The Dissertation Committee for Noah Manus Daniels

certifies that this is the approved version of the following dissertation:

Remote Homology Detection in Proteins Using Graphical Models

Committee:

Prof. Lenore Cowen, Supervisor

Prof. Donna Slonim

Prof. Benjamin Hescott

Prof. Bonnie Berger

Prof. Yu-Shan Lin

Remote Homology Detection in Proteins Using Graphical Models

A dissertation

submitted by

Noah Manus Daniels, B.S., M.S.

In partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

in

Computer Science

TUFTS UNIVERSITY

April 2013

Advisor: Prof. Lenore Cowen

For my grandmother

Acknowledgments

The work described in this dissertation is the result of the advice, contributions, and collaborations of many friends and colleagues.

First, I would like to thank my advisor, Professor Lenore Cowen, for many years of mentorship. Her encouragement, patience, and motivation have been essential to this work. I thank my dissertation committee members, Professor Donna Slonim, Professor Benjamin Hescott, Professor Yu-Shan Lin, and Professor Bonnie Berger.

My collaborators during the course of my doctoral work have also been inspiring. I thank Anoop Kumar, Matt Menke, Raghavendra Hosur, Shilpa Nadimpalli, Andrew Gallant, Po-Ru Loh, Michael Baym, Jian Peng, Jisoo Park, and Mengfei Cao for their contributions, be they in code, conversation, or collegiality.

I thank Professor Norman Ramsey, Professor Sinaia Nathanson, and Dean Lynne Pepall, along with my GIFT colleagues, for teaching me how to teach. I also thank my teaching assistants, Sarah Nolet, Joel Greenberg, Andrew Pellegrini, and Michael Pietras, for helping me teach, Nathan Ricci for the constant feedback and guest lectures, and Professors Carla Brodley and Diane Souvaine for the opportunities.

I thank the Tufts University Computer Science Department, especially Gail Fitzgerald, Jeannine Vangelist, and Donna Cirelli, for so much support over so many years.

I owe a debt of gratitude to Michael Bauer, Erik Patton, Jon Frederick, George Preble, and Eric Berg for keeping our systems running despite my best efforts to the contrary.

Much of the material in Chapter 2 of this dissertation has been published as

"Touring Protein Space with Matt", with Anoop Kumar, Matt Menke, and Lenore Cowen, in the journal *ACM Transactions on Computational Biology and Bioinformatics.* Much of the material in Chapter 3 of this dissertation has appeared as "SMURFLite: combining simplified Markov random fields with simulated evolution improves remote homology detection for beta-structural proteins into the twilight zone", with Raghavendra Hosur, Bonnie Berger, and Lenore Cowen, in the journal *Bioinformatics.* Some of the material in Chapter 4 of this dissertation has appeared as an experience report, "Experience Report: Haskell in Computational Biology", with Andrew Gallant and Norman Ramsey, in the Proceedings of the *International Conference on Functional Programming.*

Finally, I would not have reached this point without the love and support of my parents, Anne and Norman Daniels, and my wife, Rachel Daniels.

Noah Manus Daniels

TUFTS UNIVERSITY

April 2013

Remote Homology Detection in Proteins Using Graphical Models

Noah Manus Daniels

Advisor: Prof. Lenore Cowen

Given the amino acid sequence of a protein, researchers often infer its structure and function by finding homologous, or evolutionarily-related, proteins of known structure and function. Since structure is typically more conserved than sequence over long evolutionary distances, recognizing remote protein homologs from their sequence poses a challenge.

We first consider all proteins of known three-dimensional structure, and explore how they cluster according to different levels of homology. An automatic computational method reasonably approximates a human-curated hierarchical organization of proteins according to their degree of homology.

Next, we return to homology prediction, based only on the one-dimensional amino acid sequence of a protein. Menke, Berger, and Cowen proposed a Markov random field model to predict remote homology for beta-structural proteins, but their formulation was computationally intractable on many beta-strand topologies.

We show two different approaches to approximate this random field, both of which make it computationally tractable, for the first time, on all protein folds. One method simplifies the random field itself, while the other retains the full random field, but approximates the solution through stochastic search. Both methods achieve improvements over the state of the art in remote homology detection for beta-structural protein folds.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Proteins

Proteins are the molecular machines that are essential to the process of life. For example, transmembrane proteins allow molecules to move into and out of the cell. Hemoglobin ferries iron through the blood, while immunoglobulin provides for defense against pathogens. Actin contracts our muscles, and myelin insulates our nerves.

It is well known that DNA encodes the genetic information that determines how we develop and function. Portions of this DNA are transcribed into RNA, and then a complex piece of cellular machinery called the ribosome translates this RNA into amino acids, the building blocks of proteins. Proteins are the machines for which the DNA is the blueprint. Chains of amino acids fold into intricate, low-energy forms, and these structures *d*o things.

It is the structure of a protein that allows it to perform its function, and while this structure is determined by the amino acid sequence that derives from DNA, the relationship between sequence and structure is not simple.

1.1.1 Primary Structure

Proteins are composed of linear chains of molecules called *amino acids*. An amino acid is a molecule comprising an *amine* group, a *carboxyl* group, and one of twenty

possible *sidechains* (see Figure 1.1). Each of these components is attached to a carbon atom, known as the α-carbon.

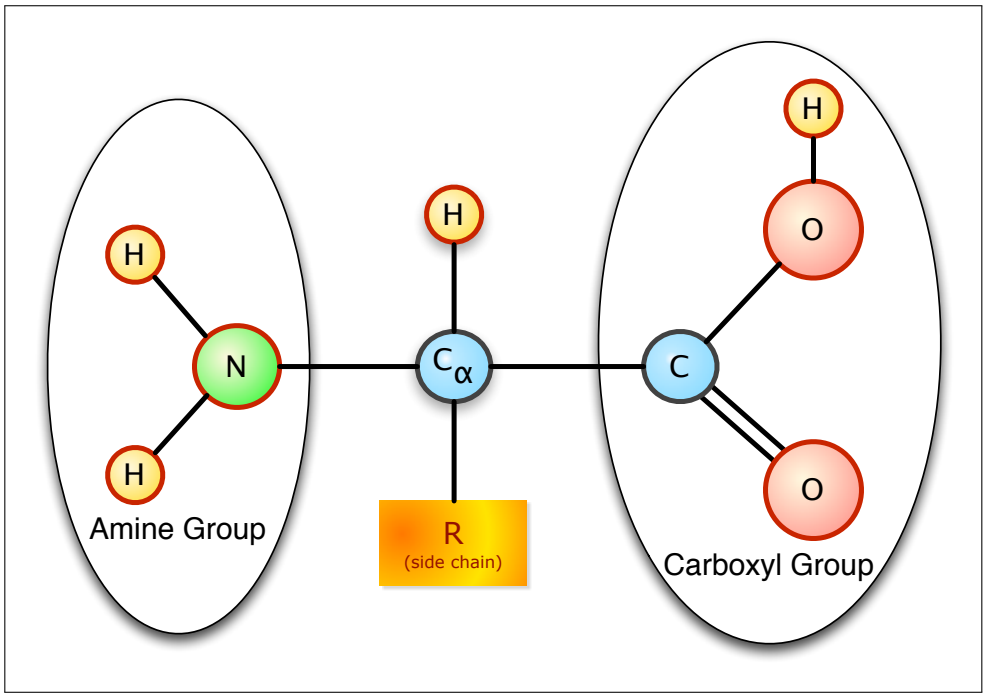


Figure 1.1: The general structure of an amino acid showing the hydrogen(H), nitrogen(N), oxygen(O) and carbon(C,C_α) atoms. The sidechain is one of twenty possible "decorations;" amino acids differ only in their sidechains.

Amino acids bind to one another via a *peptide bond*, which forms when the carboxyl group of one amino acid gives up an oxygen and hydrogen to bind with the amine group of another amino acid, which gives up a hydrogen. This results in a free water molecule. In addition, as multiple amino acids form *polypeptide chains*, the unbound amine group at one end is known as the *N-terminal* end of the resulting protein, while the unbound carboxyl group at the other end is known as the *C-terminal* end (Figure 1.2).

The peptide linkages, along with the α-carbon atoms, form the *backbone* of the protein. Ultimately, the protein folds into a globular form, generally representing a lowest-energy conformation. It is useful to describe the structure of proteins at several levels of organization.

The *primary structure* of a protein is simply its sequence of amino acids. In

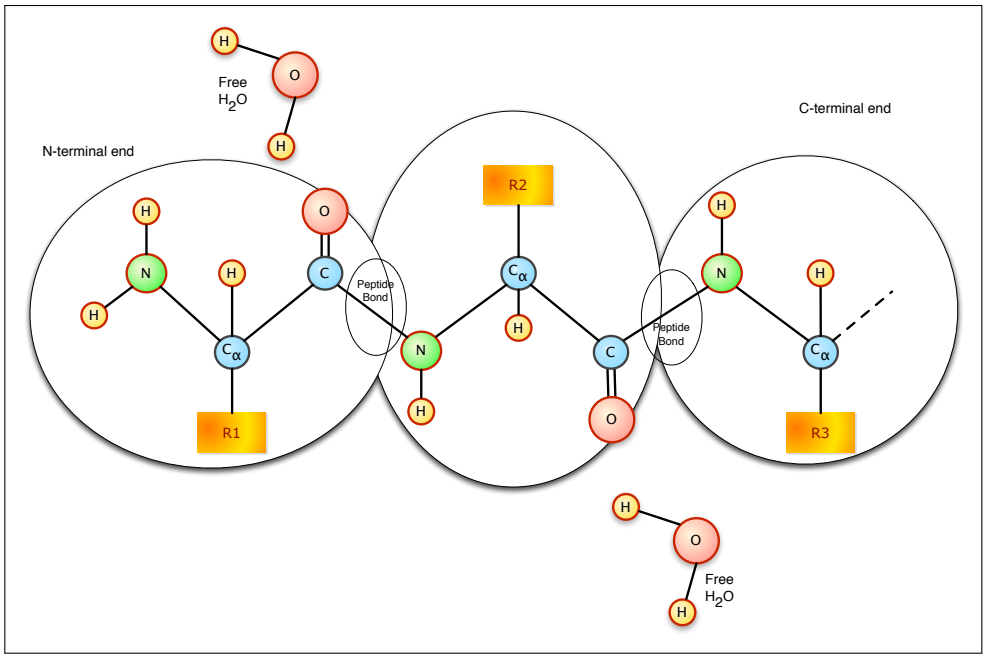


Figure 1.2: Amino acids join by peptide bond to form the backbone. Individual amino acids are highlighted with ovals. The sidechain connected to the C_α is different for each amino acid. When a peptide bond forms, a water molecule forms from the hydrogen given up by the nitrogen end of one amino acid, and the oxygen and hydrogen given up by the carbon end of the other.

principal, any of the 20 standard amino acids can occur in any position of an amino acid chain; for a protein of length n, there are 20^n possible protein sequences. Of course, the subset of those sequences that will fold into a compact, three-dimensional structure is much smaller; the subset of *those* that would fold into a compact, three-dimensional structure that exists in nature is smaller still. However, determining which protein sequences nature allows is not trivial.

1.1.2 Secondary Structure

Local interactions among amine and carbonyl groups result in *hydrogen bonds* between amino acids that are not immediately adjacent in sequence. A hydrogen bond is the electrostatic attraction between a hydrogen atom in one amino acid and an oxygen or nitrogen atom in another. In particular, we can describe the *secondary structure* of a protein according to the shape of the angles of the backbone. The most common type of secondary structure is the α-helix, in which the protein back-

bone coils into a twisted shape, stabilized by hydrogen bonds (Figure 1.3). The most common α-helices have hydrogen bonds between residues four positions apart in sequence. Other, less common helical structures include the 3_{10} helix, in which residues three apart in sequence form hydrogen bonds, and the π helix, in which residues five apart in sequence form hydrogen bonds.

Another secondary structure is the β-strand, which in combination form β-sheets. β-strands occur when the backbone is stretched out; typically, this conformation is stabilized by hydrogen bonds between adjacent strands (Figure 1.4), resulting in β-sheets. The hydrogen bonds in β-sheets may occur between residues that are very far apart from each other in the amino acid sequence. β-strands in a sheet may be parallel or anti-parallel to one another with respect to the direction of the amino acid sequence.

The remainder of local backbone conformations, consisting of turns, bulges, loops, bridges, etc., have been classified into several different subcategories, but is often grouped together into a third category of secondary structure, commonly referred to as a *coil*.

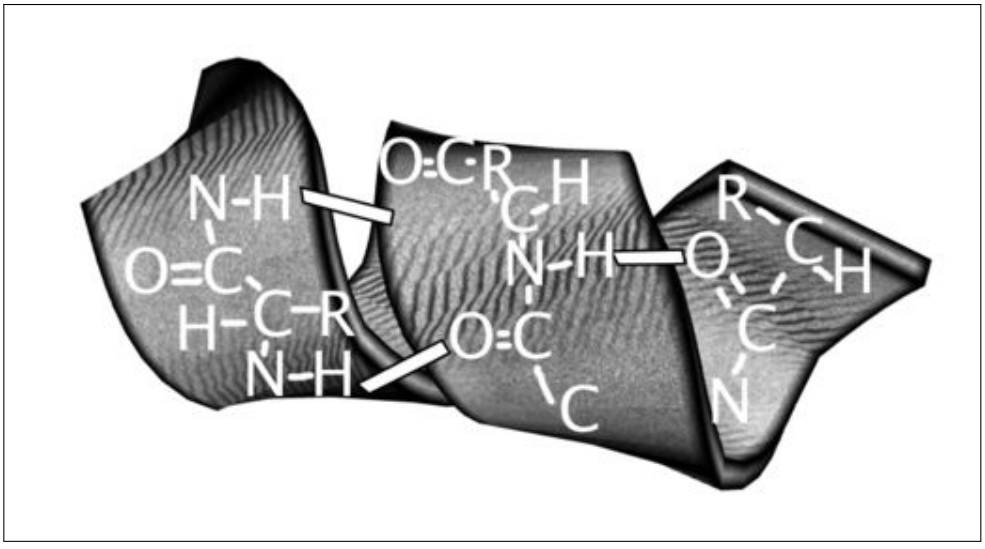


Figure 1.3: α-helix secondary structure. Hydrogen bonds between residues 4 positions apart in sequence cause the helical shape. Other, less common helix structures include the 3_{10} helix, in which residues 3 apart in sequence form hydrogen bonds, and the π helix, in which residues 5 apart in sequence form hydrogen bonds.

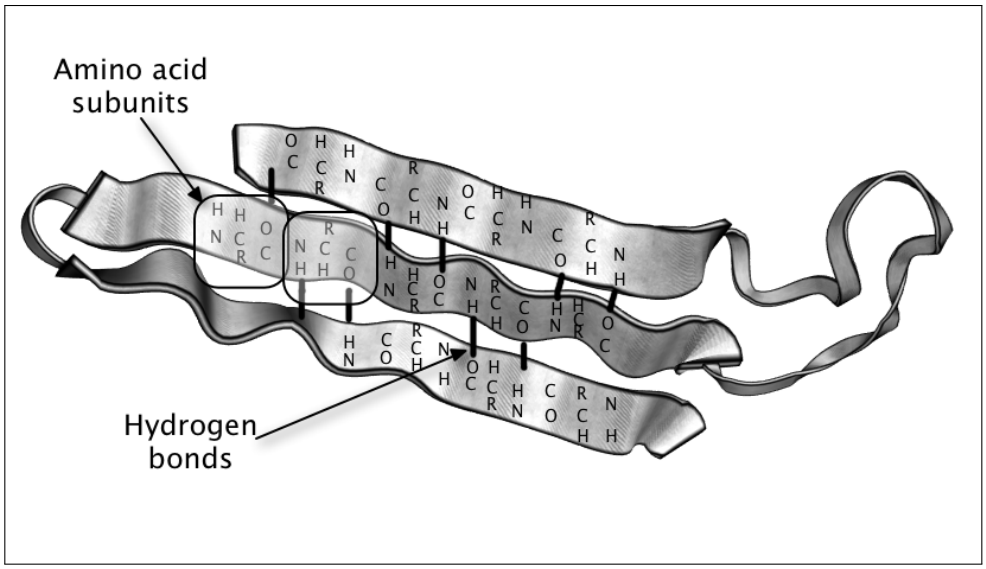


Figure 1.4: β-sheet secondary structure. Hydrogen bonds between residues that may be quite far apart in sequence cause this pleated, sheet-like shape. Antiparallel β-strands are shown here; parallel β-strands also exist.

1.1.3 Supersecondary and Tertiary Structure

We can mark secondary structural elements of the complete structure of a protein backbone as it is folded in three-dimensional space, and consider the pattern of where the α-helices and β-strands lie. For example, β-strands can be organized into β-barrels (Figure 1.5), sandwiches, or propellers; α-helices can be organized into 2- or 4-helix bundles, and there are other patterns of strand topologies that involve mixed collections of α-helices and β-strands. The topologies of the various strand positions are known as *super-secondary structure*.

The *tertiary structure* of a protein is the fully-specified three-dimensional position of every atom. The orientation of the backbone atoms in three-dimensional space forms three distinguishing dihedral angles: ϕ between the carbon-1-nitrogen and α-carbon-carbon-1 atoms in an amino acid, ψ between the nitrogen-α-carbon and carbon-1-nitrogen atoms, and ω between the α-carbon-carbon-1 and the nitrogen and α-carbon of the next amino acid (see Figure 1.6). The angle ω is usually $0°$, and occasionally $180°$. The side chain of each amino acid must then pack into a low-energy state in such a way that it does not interfere with the other amino

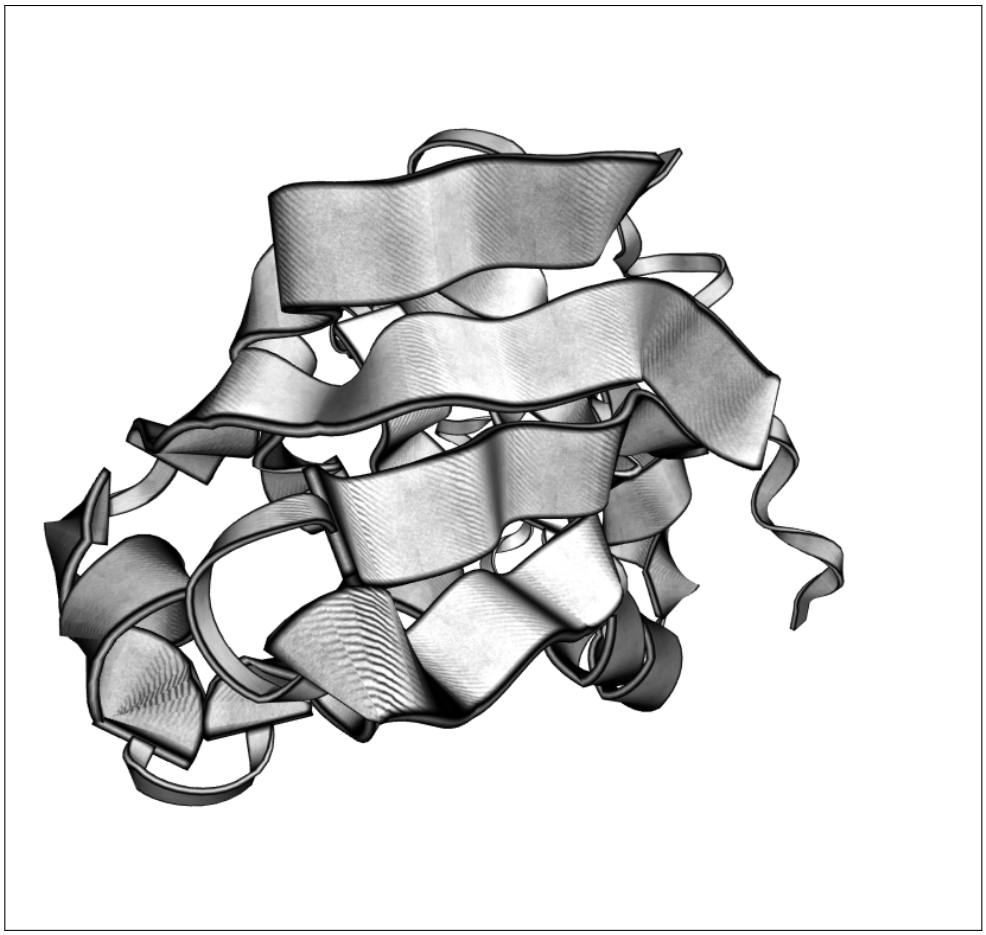


Figure 1.5: Super-secondary structure "cartoon" of Barwin (PDB ID 1BW3). Barwin, an endoglucanase, has eight β-strands forming a closed "barrel" shape, as well as four α-helices.

acids in the protein. The tertiary structure represents, in most cases, a global minimum energy state, also known as the *native state*. Many proteins have now had their tertiary structure determined by X-ray crystallography, or by nuclear magnetic resonance (NMR) spectroscopy. A protein whose structure has been determined experimentally is said to have a *solved* structure. However, the difficulty of experimentally solving the structure of any particular protein of interest can vary. X-ray crystallography's limiting factor is that not all proteins can be put into solution and crystallized, while NMR's limiting factor is primarily computational. The Protein Data Bank (PDB) [BKW$^+$77, BBB$^+$00] is a publicly available database that contains the atomic coordinates of all proteins whose tertiary structure has been

solved.

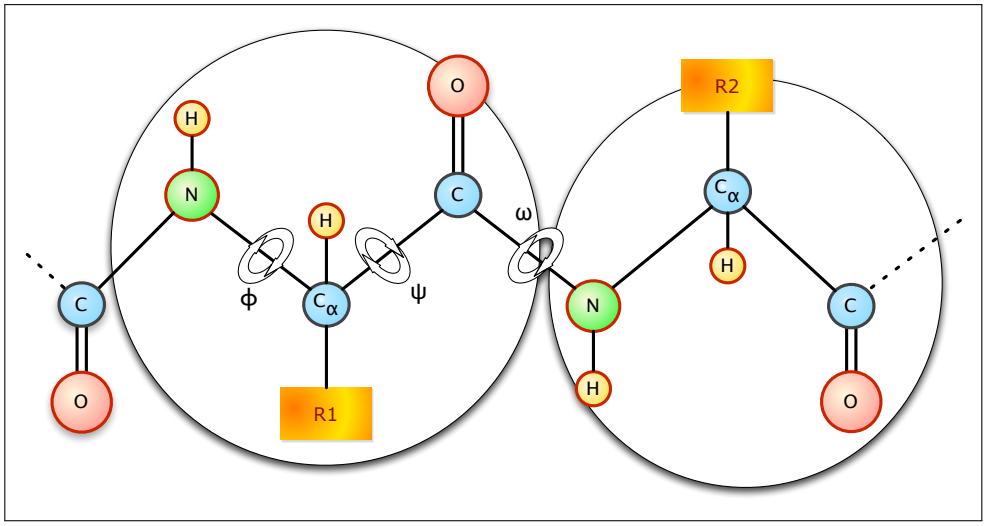


Figure 1.6: The orientation of the backbone atoms in three-dimensional space forms three dihedral angles: ϕ between the carbon-1-nitrogen and α-carbon-carbon-1 atoms, ψ between the nitrogen-α-carbon and carbon-1-nitrogen atoms, and ω between the α-carbon-carbon-1 and the nitrogen and α-carbon of the next amino acid.

Finally, *quaternary structure* describes how multiple tertiary structures interact; these may be multiple duplicate protein chains (for example, a homodimer is a complex of two identical protein chains, while a heterodimer is a complex comprising two different protein chains). In this work, we focus on individual chains, rather than quaternary structures.

1.1.4 Protein Data Sets

In order to make sense of the evolutionary, structural, and functional relationships among proteins, biologists have created several organizational schemes. Structural Classification Of Proteins (SCOP) [MBH95, AHB+04] and CATH (which stands for Class, Architecture, Topology, and Homologous superfamily) [OMJ+97, PBB+03, GLA+07] are hierarchical schemes that place proteins in a tree based primarily on structural, but also on evolutionary and functional similarities. In this work, we will primarily rely on SCOP, since it has been used in many homology detection studies [ES99, WS04, Söd05].

SCOP organizes all protein sequences of known structure (with some time delay) into a four-level hierarchy. The top level of the SCOP hierarchy is *class*, which distinguishes the primary secondary-structural composition of proteins: mainly-α, mainly-β, mixed α and β, cellular-membrane proteins, among others. The second level of the SCOP hierarchy is *fold*, which organizes proteins by overall structural motif, or supersecondary structure. Proteins in the same fold are not necessarily evolutionarily related. Below fold is *superfamily*, which organizes proteins that share evolutionary relationships, as well as similar structure and function. Below the superfamily level is the *family* level of SCOP. Proteins in the same family have clear evolutionary relationships, and a significant level of sequence similarity. Figure 1.7 illustrates the SCOP hierarchy.

1.1.5 Protein Folding

The process by which a protein, as its amino acid sequence is emitted by the ribosome, forms its stable tertiary structure is called *folding*. In 1969, molecular biologist Cyrus Levinthal noted [Lev69] that even given a coarse (tripartite) discretization of bond angles, a protein of merely 100 amino acids (a short chain by most standards) could take 3^{300} distinct three-dimensional conformations (tertiary structures). Given the accepted view that proteins typically fold to globally minimum energy states, Levinthal noted that a protein would take the lifetime of the universe to find a minimum energy state by sampling the entire fold space. However, in practice, proteins fold in microseconds or milliseconds. This apparent paradox became known as Levinthal's Paradox; the solution to the paradox must be that nature does not explore the entire fold space.

The rapidity of protein folding is thought to be because proteins fold along *folding funnels*, which prune much of the possible fold space very quickly [DC97, MCBR98, TKMN99, DSSD00]. It is even conjectured [Ros02, CSL$^+$09] that only those proteins that exhibit fold funnels that allow them to fold quickly have evolved; protein sequences that would not quickly find stable native states would be selected against during the course of evolution.

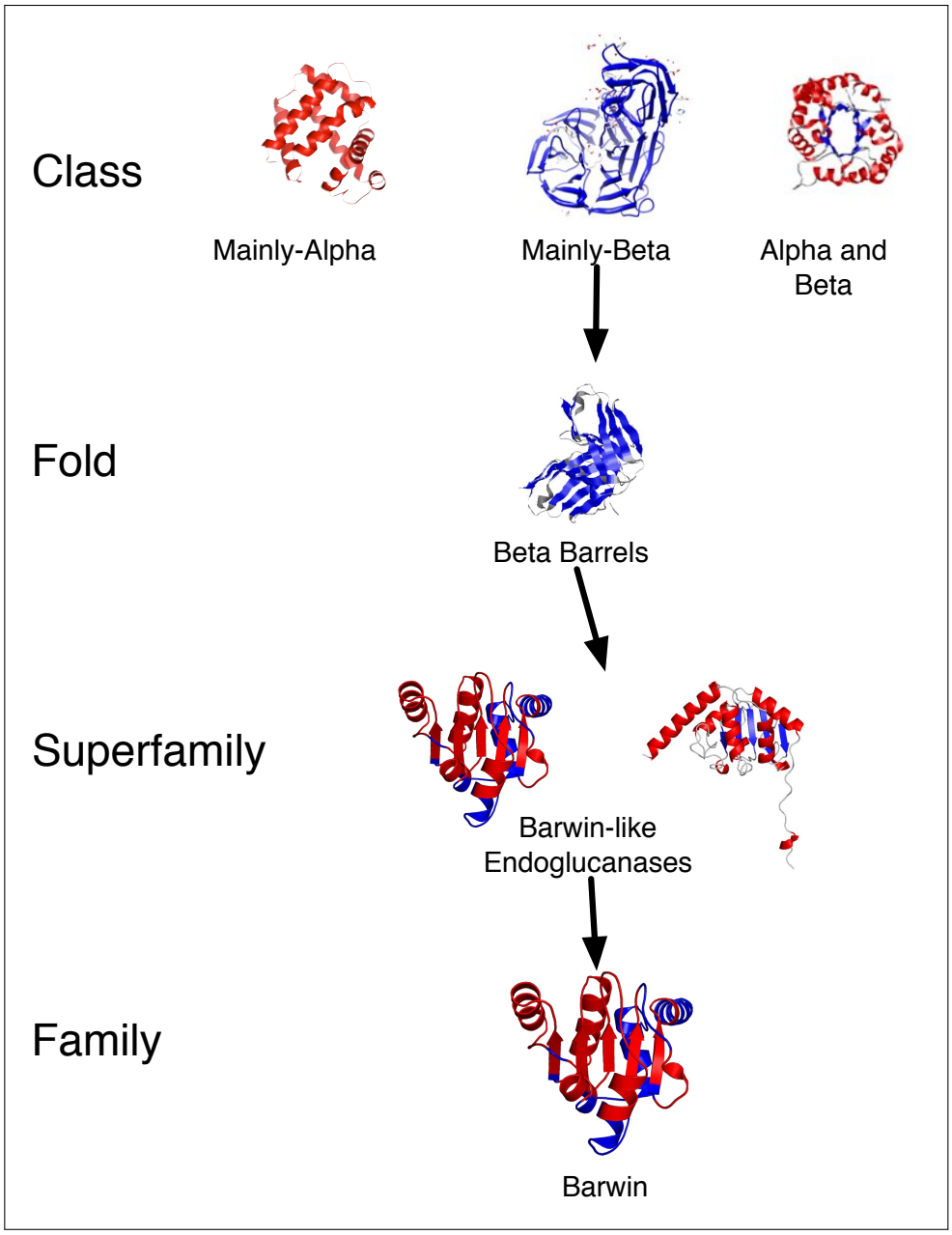


Figure 1.7: The SCOP hierarchy of protein structure. *Class* organizes proteins in large part according to supersecondary structural content. *Fold* organizes proteins by supersecondary structural motifs. *Superfamily* organizes proteins by structural, functional, and evolutionary similarity, while *Family* organizes them by sequence similarity as well.

Physics-based approaches to computationally solving the protein folding problem try to solve the various force field equations (including hydrophobic, electrostatic, and van der Waals forces) to find a minimum-energy state [BBC99, Heu99].

However, even the most simplified models can prove computationally intractable. In 1998, Berger and Leighton [BL98] proved that the seemingly simple HP (hydrophobic-hydrophilic) lattice model of protein folding is NP-hard.

For some purposes, such as understanding the molecular motion of proteins such as ion channels (which control the flow of ions through a cellular membrane) or flagellin (which forms the moving filament in bacterial flagella), just knowing the native state is not enough, and molecular dynamics simulations are necessary.

The current state of the art in full tertiary structure prediction via molecular dynamics modeling relies on huge computational infrastructures; we describe two of them. The first, Anton, is a supercomputer purpose-built for protein simulations by the D.E. Shaw Research [SDD+07]. The second, Folding@home, is a worldwide distributed-computing system developed at Stanford [JVP06]; it uses spare CPU and GPU cycles on desktop, laptop, and video game systems around the world. Both of these systems can compute a few milliseconds of simulation time per day.

Fortunately, however, it is not always necessary to determine tertiary structure to the level of precision achieved by experimental methods. Computational biology methods that use statistical energy functions and secondary or supersecondary structure prediction have made significant progress in the last ten years [Mou06]. In particular, *approximately* predicting the tertiary structure–or predicting the supersecondary structure–may be adequate when the end goal is function prediction or homology detection.

1.2 Protein Homology

An alternative to experimentally predicting the structure of a protein is to try to determine, based on *sequence* similarity, that a protein of interest is sufficiently closely related, in evolutionary terms, to some other protein of solved structure that it is likely to fold into a similar shape. However, while the task gets easier the closer it becomes to that of determining sequence, the quality of the results worsens; protein sequence is less well *conserved* than structure; that is, fairly sig-

nificantly different protein sequences may nonetheless share quite similar structures and functions [Dun06].

Biologists say that two proteins are *homologous* when they are derived from a common ancestor. Often, homologous proteins share common structure. When two protein sequences are similar, it is relatively easy to determine that they are homologous. However, homologous proteins may differ significantly in terms of sequence identity. Sequence analysis methods have long allowed for the detection of homologous proteins, provided sequence divergence is not too great. The problem of detecting homologous proteins when sequence similarity is low is known as *remote* homolog detection. The purpose of this thesis is to develop novel methods for remote homology detection. Now, we will survey existing methods for homology detection.

1.2.0.1 BLAST

Altschul, et al. developed the Basic Local Alignment Search Tool (BLAST) [AGM+90] algorithm as a faster alternative to dynamic programming-based methods such as the Smith-Waterman [SW81] algorithm. BLAST uses a number of heuristics to reduce the time required to perform an alignment, at the possible expense of some accuracy. BLAST also relies on an indexed database of sequences to be searched.

BLAST allows for fast search through databases to find potential homologs. The protein-specific version of BLAST is called BLASTP. BLASTP uses a *substitution matrix* to score alignments; the most commonly used substitution matrix is BLOcks of amino acid SUbstitution Matrix (BLOSUM) [HH92]. A BLOSUM score $s(i, j)$ for two residues i and j is given by:

$$s(i,j) = \frac{log\frac{P_{i,j}}{f_i f_j}}{\lambda} \tag{1.1}$$

where $P_{i,j}$ is the probability of observing residues i and j aligned in homologous sequences, and f_i is the observed background frequency of residue i, and λ is a scaling factor chosen to produce integer values for the scores [HH92].

Different variants of the BLOSUM matrices exist; for a chosen threshold

L, only sequences within a sequence identity threshold of $L\%$ are clustered into a single representative sequence; those sequences are then aligned and the alignment used to compute the BLOSUML matrix. Thus, BLOSUM80 is intended for use in less divergent sequence alignments, while BLOSUM50 is intended for use in more divergent sequence alignments. BLOSUM62 is a commonly used default scoring matrix for protein sequence alignment tools such as BLAST [AMS+97].

There are newer BLASTP variants, such as PSI-BLAST [AMS+97] and DELTA-BLAST [BSA+12] that improve sensitivity by replacing BLOSUM with a *Position-specific scoring matrix* (PSSM) that scores mismatches differently depending on where they occur in the alignment. PSI-BLAST determines its PSSM by iterative search: First, it performs a standard BLASTP search, and computes a PSSM from the resulting alignment. It then repeats this process, searching with the PSSM created by the previous iteration, and computing a new PSSM. In contrast, DELTA-BLAST uses pre-determined PSSMs derived from the Conserved Domains Database (CDD) [MBA05], essentially groups of proteins already determined to be homologous.

BLAST and its derivatives, such as PSI-BLAST and DELTA-BLAST, are effective at identifying homologous protein sequences for a query sequence when those homologous sequences share a reasonable amount of sequence identity with the query sequence [Ros99]. However, we wish to be able to identify homologous proteins–those that share structural, functional, and evolutionary relationships–even when they do not share a great deal of sequence similarity. Since protein structure is more highly conserved than sequence [DBR97], we would like to incorporate information that is not simply derived from sequence alignments.

1.2.1 Structural Alignment

Just as we can align the sequences of two or more proteins in order to compare them, we can also align the *structures* of two or more proteins. Clearly, protein structure alignment requires knowing the tertiary structure–the three dimensional coordinates of all the atoms, or at very least the backbone atoms–of the proteins to

be aligned.

Structural alignment can be used to measure structural similarity, and from there infer functional and evolutionary relationships. Structural alignment can also be used to measure the quality of a computational protein structure *prediction* versus a known, solved structure. In general, protein structural alignment relies on some form of geometric superposition, though a wide variety of algorithms exist for efficiently computing this superposition. In fact, computing the *optimal* geometric superposition is known to be NP-hard [WJ94]. Several heuristic approaches have been developed for practical protein structural alignment. DALI [SB98], for example, breaks the structures into hexapeptide fragments and calculates a distance matrix by evaluating the contact patterns between fragments. DALI then compares these distance matrices and applies a score-maximization search to compute an alignment. This approach is called "aligned fragment pair" alignment, and is also used by MAMMOTH [OSO09], which instead applies dynamic programming to compute the alignment. Matt [MBC08] also uses aligned fragment pairs, but allows "impossible" translations and twists in order to better capture structural similarities at the superfamily or even fold levels. Hybrid aligners, which use both sequence and structure information, also exist. DeepAlign [Wan12] incorporates not just atomic coordinates but also secondary structural annotation and sequence information. Our own Formatt [DNC12] also combines sequence and structure information, to try to avoid "register errors" that trade significant sequence alignment errors for small structural gains. Most of these methods are fundamentally solving a bi-criterion optimization problem: we wish to align as much of the input proteins' structure as possible, while at the same time minimizing the root mean square distance (RMSD) of the resulting alignment, where RMSD is defined as the square root of the average distance between corresponding α-carbon atoms between the backbones of the proteins in alignment [Kab76].

When aligning more distantly-related proteins, structural alignment methods often outperform purely sequence-based methods [CL86]. For this reason, protein structural alignment is routinely used to produce alignments of homologous proteins

to form training sets for remote homology detection techniques.

1.3 Hidden Markov Models

A Markov model represents a series of observations via a probabilistic finite-state automaton. A Markov model on an alphabet A is a triplet

$$M = (Q, \pi, \alpha), \tag{1.2}$$

where Q is a finite set of states, each state generates a character from A, π is the set of initial state probabilities, and α is the set of state transition probabilities. Markov models are so named because they uphold the *Markov property*, which states that future states depend only on the current state of the system. In other words, first-order Markov models are "memoryless." A Markov model may take into account a *fixed* number of past states (a k_{th}-order Markov model make take into account k past states).

A hidden Markov model (HMM) represents a series (sometimes a time series) of observations by a "hidden" stochastic process. HMMs were originally developed for speech recognition [Vit67, Rab89]. A HMM is on an alphabet A is a 5-tuple

$$M = (Q, V, \pi, \alpha, \beta), \tag{1.3}$$

where Q is again a finite set of states, V is a finite set of observations per state, π is the set of initial state probabilities, α is again the set of state transition probabilities, and β is the finite set of emission probabilities over the alphabet A. Hidden Markov models differ from ordinary Markov models in that, while the emissions are observable, the states occupied by the finite state machine are not themselves observable. There are three problems to be solved regarding HMMs, and correspondingly, three algorithms to solve them.

The first problem is: *Given an HMM M and an observed sequence S, compute the most probable path through M that generates S.*

This problem is solved by the *Viterbi algorithm* [Vit67], which is typically implemented using dynamic programming. The Viterbi algorithm solves the recurrence relation:

$$\begin{aligned} V_{1,k} &= \mathrm{P}\big(s_1 \mid k\big) \cdot \pi_k \\ V_{t,k} &= \mathrm{P}\big(s_t \mid k\big) \cdot \max_{x \in Q} \left(a_{x,k} \cdot V_{t-1,x}\right) \end{aligned} \tag{1.4}$$

where $V_{t,k}$ is the probability of the most probable state sequence emitting the first t observations with k as its final state, and $s_i \in S$ is the i^{th} observation in S. The corresponding path through the model can be retrieved by remembering what series of transitions among states $x \in Q$ were chosen when solving the recurrence relation.

The second problem is: *Given an HMM M and a sequence of observations S, compute $P(S|M)$, the probability of observing the sequence S emitted by the model M.*

This problem is solved by the *forward algorithm*, which relies on dynamic programming as well. In essence, the forward algorithm sums the probabilities over all possible state paths that can emit S. The recurrence relation for the forward algorithm is nearly identical to that for the Viterbi algorithm, except that it sums, rather than choosing the maximum from, the probabilities at each step:

$$\begin{aligned} V_{1,k} &= \mathrm{P}\big(s_1 \mid k\big) \cdot \pi_k \\ V_{t,k} &= \mathrm{P}\big(s_t \mid k\big) \cdot \sum_{x \in Q} \left(a_{x,k} \cdot V_{t-1,x}\right) \end{aligned} \tag{1.5}$$

The third problem is: *Given a set of sequences of observations, O, and a model M, determine the transition probabilities α and emission probabilities β that maximize the $P(O|M)$, the likelihood of observing the set of sequences given the model.*

Typically, a solution to this problem is *estimated* by the *Baum-Welch algorithm* [BPSW70], which is an expectation-maximization algorithm. A more computationally efficient but less accurate alternative is the Viterbi Training algorithm (not to be confused with the Viterbi algorithm), also known as *segmental*

k-means [Rab89]. A simulated annealing search approach to Baum-Welch can also be used to avoid local optima [BCHM94].

A further explanation of the above algorithms can be found in [Rab89].

Despite their origins in the field of speech recognition, hidden Markov models have been used in a variety of areas within the realm of computational biology. In the context of DNA sequence analysis, HMMs have been used [DLC02] to detect "CpG islands," regions of the genome where cytosine and guanine are predominant and adjacent in sequence. CpG islands are useful for determining the start of transcription sequences–the markers that indicate the regions of the genome that code for protein sequences. Hidden Markov models were first used to search for DNA sequences in genome databases by Churchill [Chu89] in the late 1980s. Later, Krogh et al. [KBM+94] used HMMs to model protein evolution.

1.3.1 Profile Hidden Markov Models

With respect to homology detection, *profile* hidden Markov models have been popular. In particular, profile HMMs have been used to model families of protein sequences, in order to predict whether newly-discovered sequences belong to those families. Profile hidden Markov models attempt to represent the evolutionary processes underlying the differences among closely-related proteins. In addition, HMM-derived clusterings of proteins have been published, such as Pfam [FMSB+06], PROSITE [HBB+06], and SUPERFAMILY [WMV+07].

HMMER [Edd98] and SAM [HK96] are two popular software tools for homology detection in proteins (though both are also widely used in nucleotide sequence analysis, as well). Much of the work in this dissertation is based on HMMER; we chose it as it is open-source and more actively maintained.

HMMER models three types of events that may occur during the evolution of a protein: *insertion*, *deletion*, and *substitution* of an amino acid at a particular position. These three possible events become the three hidden states of the HMM. Substitution events are modeled using a *match* state, which also represents amino acids that are conserved, or have not changed, between proteins. In essence, mu-

tated amino acids can be represented as substitutions using a substitution matrix, and since the most probable substitution in such a matrix is the identity function, conserved amino acids can also be represented using the same matrix. Insertion and match states are both considered *emission* states, as each corresponds to the presence of an amino acid at a particular position in a protein. Each emission state comprises a table of emission probabilities: the likelihood that any particular amino acid will be present (emitted) at that position. Intuitively, for each match state, the most common amino acid seen in the training data will be the most probable amino acid in the emission table for that column of the alignment.

HMMER uses the "Plan7" hidden Markov model architecture, which forbids direct transitions between insertion states and deletion states [Edd98]. "Plan7" is a pun on "Plan9," the architecture by Krogh, et al. [KBM+94] that allowed all 9 possible transitions among match, insert, and delete states; "Plan7" gets its name because there are exactly 7 possible transitions into the states of any column of the alignment used for training. See Figure 1.8 for an illustration of the Plan7 architecture.

HMMER trains a profile HMM (using a simulated annealing variant of the Baum-Welch algorithm) [MSE96] on a *sequence profile*, which is an alignment of the protein sequences comprising some group–such as a SCOP superfamily or family–of putatively homologous proteins. This alignment may be a sequence alignment or a structural alignment; in this work we will focus on profiles derived from structural alignments.

An alignment used for training may of course contain *gaps*. A gap in row 2, column j indicates that as proteins evolved, either protein 2 lost its amino acid in position j, or other proteins gained an amino acid in position j. If column j contains few gaps, it is considered a *consensus column*, and the few proteins with gaps may have lost amino acids via *deletions*. Note that this model is directionless with respect to evolutionary change; it does not distinguish between a residue being gained or lost over time. If column j contains *mostly* gaps, it is considered a *non-consensus column*, and the few proteins without gaps may have gained amino acids

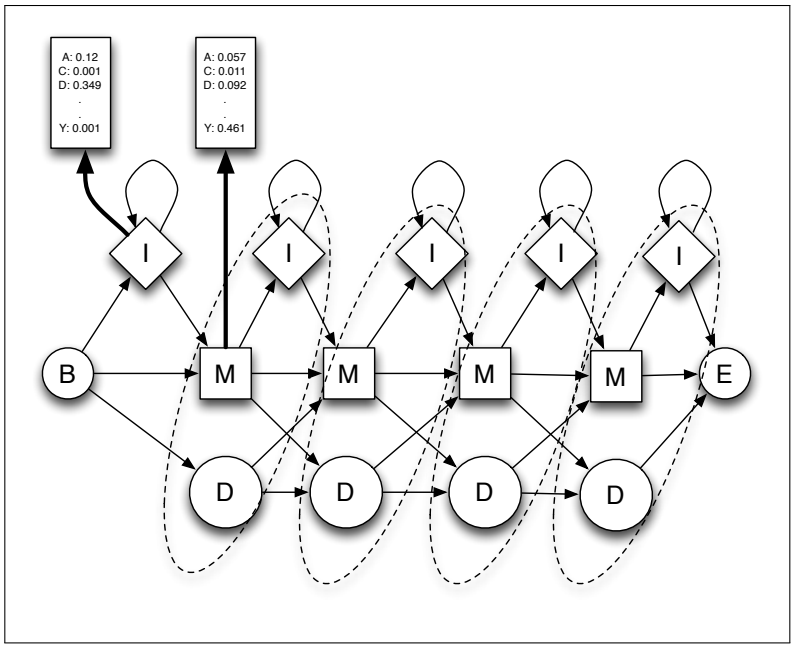


Figure 1.8: The "Plan7" architecture for hidden Markov models, as implemented in HMMER. Dashed circles indicate *nodes* of the model. A node groups a match, insertion, and deletion state, along with the emission probabilities for the match and insertion states. Note: this diagram simplifies the "Plan7" architecture; in reality, begin and end nodes are more complex, allowing for entire models to repeat.

via *insertions*.

We refer to the amino acid sequence of a protein whose structure we do not know, and wish to determine using homology detection, as a *query sequence*. Homology detection using a hidden Markov model involves *aligning* a query sequence to a hidden Markov model, or computing a *path* through the model that maximizes the likelihood of the model emitting the query sequence. This alignment involves assigning successive amino acids in the query sequence to successive nodes of the model. For a given node of the model, the match and deletion states are mutually exclusive, as are the insertion and deletion states. However, it is permissible for a path to assign amino acids to both the match and insert states of a node. In addition, the match state consumes exactly one amino acid from the query sequence, while the insert state may consume many. The delete state consumes no amino acids from the query sequence.

Given a hidden Markov model, a protein whose query sequence has a higher

probability is considered to be more likely to be homologous to the proteins in the alignment. We write a query sequence as $x_1, \ldots, x_N$, where each x_i is an amino acid. The number of amino acids, N, can differ from the number of columns in the alignment, C.

A hidden Markov model carries emission probabilities on some states, and transition probabilities on all edges between states. Both the probabilities and the states are determined by the alignment:

- For each column j of the alignment, the hidden Markov model has a *match state* M_j. The match state contains a table $e_{M_j}(x)$ which gives the probability that a homologous protein has amino acid x in column j.

- For each column j of the alignment, the hidden Markov model has an *insertion state* I_j. The insertion state contains a table $e_{I_j}(x)$ which represents the probability that a homologous protein has gained amino acid x by insertion at column j.

- For each column j of the alignment, the hidden Markov model has a *deletion state* D_j. The deletion state represents the probability that a homologous protein has lost an amino acid by deletion from column j.

The probabilities $e_{M_j}(x)$ and $e_{I_j}(x)$ are *emission probabilities*. Each tuple of match, insertion, and deletion states is called a *node* of the hidden Markov model.

Each transition has its own probability:

- A transition into a match state is more likely when column j is a consensus column. Depending on the predecessor state, the probability of such a transition is $a_{M_{j-1}M_j}$, $a_{I_{j-1}M_j}$, or $a_{D_{j-1}M_j}$.

- A transition into a deletion state is more likely when column j is a non-consensus column. The probability of such a transition is $a_{M_{j-1}D_j}$ or $a_{D_{j-1}D_j}$.

- A transition into an insertion state is more likely when column j is a non-consensus column. The probability of such a transition is $a_{M_{j-1}I_j}$ or $a_{I_{j-1}I_j}$.

Due to the specific topology of the state-transition graph in the "Plan7" architecture, a reformulation of the Viterbi recurrence relations are warranted. In particular, we need not consider all state transitions that would be possible given a general topology, and instead, need consider only three possible transitions at each node, which reduces the search space. The variant of the Viterbi algorithm adapted for the "Plan7" architecture is given by the recurrence relations:

$$V_j^M(i) = \frac{e_{M_j}(x_i)}{q_{x_i}} \times \max \begin{cases} V_{j-1}^M(i-1) \times a_{M_{j-1}M_j} \\ V_{j-1}^I(i-1) \times a_{I_{j-1}M_j} \\ V_{j-1}^D(i-1) \times a_{D_{j-1}M_j} \end{cases}$$

$$V_j^I(i) = \log \frac{e_{I_j}(x_i)}{q_{x_i}} \times \max \begin{cases} V_j^M(i-1) \times a_{M_j I_j} \\ V_j^I(i-1) \times a_{I_j I_j} \end{cases} \tag{1.6}$$

$$V_j^D(i) = \max \begin{cases} V_{j-1}^M(i) \times a_{M_{j-1}D_j} \\ V_{j-1}^D(i) \times a_{D_{j-1}D_j} \end{cases}$$

1.4 Other Homology Detection Methods

1.4.1 Threading Methods

Threading is a methodology by which a query sequence is *threaded* onto a structural template, and the quality of the threading is evaluated by means of an energy function or a statistical likelihood. The idea of threading is based on the observation that the number of unique protein folds found in nature is small with respect to the number of distinct protein sequences, and that relatively few novel protein folds have been found recently [PBB+03].

THREADER [JTT92], the original threading approach, aligns a protein sequence to a full tertiary structure model of a protein, and computes a score based upon a Boltzmann energy function and solvent potentials. THREADER thus eval-

uates the propensity a sequence has for forming a particular tertiary structure, but it cannot distinguish homologs (evolutionarily related proteins that share structure and possibly function) from *analogs* (proteins that happen to share similar structure but have no evolutionary relationship) [OJT94, Jon97].

GenTHREADER [Jon99] improves upon THREADER, using an artificial neural network to compute a score based upon multiple inputs: solvent and Boltzmann potentials like THREADER, but also a sequence alignment score and length, and the lengths of the query sequence and template.

Another popular and successful threading method is RAPTOR [XLKX03], which relies on a template based on a *contact map* to indicate which residues in a protein are in close geometric proximity to one another, as well as the statistical propensities for individual residues to be in such proximity. RAPTOR then relies on linear programming to compute the optimal alignment of a query sequence to this template, in order to minimize the statistical energy. In Chapter 3, we compare our results for remote homology detection to those of RAPTOR.

Other threading methods include SPARKS X [YFZZ11] and the recently-developed RaptorX [PX11b].

1.4.2 Profile-Profile Hidden Markov Models

Several recent efforts have improved upon profile hidden Markov models, by aligning a profile HMM built from a training profile (much like HMMER) with another profile HMM built from the query sequence. HHPred [Söd05], MUSTER [WZ08], and HHblits [RBHS12] are three such approaches. Given a query sequence, HHPred relies on PSI-BLAST [AMS$^+$97] to build a sequence profile. HHPred then builds a profile HMM on this profile, and uses a variant of the Viterbi algorithm to align the *query* HMM to candidate *target* HMMs. HHPred, along with other profile-profile hidden Markov model methods, relies on the query profile to more faithfully represent the evolutionary variation in the protein sequences that may be homologous to the query sequence. In Chapter 3, we compare our results for remote homology detection to those of HHPred.

1.4.3 Markov random fields

Some researchers have suggested generalizing HMMs to the more powerful Markov random fields (MRFs). Unlike HMMs, which model only local dependencies among neighboring residues, MRFs can capture nonlocal interactions, such as the conservation of hydrogen-bonded residues in paired β-strands. SMURF (Structural Motifs Using Random Fields) [Men09, MBC10] used this β-strand information to recognize remote homologs in the β-propeller folds better than HMM methods. However, SMURF is limited by computational complexity, because it uses multidimensional dynamic programming to compute an optimal parse of a query sequence onto the MRF, and its computational complexity is exponential in something called the interleave number of a structure. This interleave number is simply the number of intervening β-strands (in sequence) between a pair of hydrogen-bonded, paired β-strands. β-propellers have a maximum interleave number of three, and thus they are tractable for SMURF. In contrast, some β-barrels and sandwiches have an interleave number as high as 12, and thus, SMURF's computational complexity becomes intractable on available computer systems. Chapters 3 and 4 explore two alternative approaches to mitigate this computational hindrance.

1.5 Remote Homology Detection

While computing tertiary structure is computationally challenging, we can take comfort in the fact that we do not always need tertiary structure to make useful predictions as to function or evolutionary similarity. In particular, supersecondary structure is often enough. Since structure determines function, if we can classify a new protein of unknown structure as sharing similar supersecondary structure to a group of known proteins, we have evidence that the new protein shares a similar function to those known proteins.

Protein sequence is much typically less conserved than structure [KPH06, WKG00], so proteins of similar structure and function, as seen at the SCOP superfamily level, may lack any meaningful sequence similarity. Simple sequence compar-

isons such as BLAST fail to correctly identify these remote homologs. However, if biologists sequence the genome of an organism, they will wish to functionally annotate the proteins coded for by its genes. Anton or Folding@Home would require weeks or months of computational time per gene to compute tertiary structure, and even a bacterium has thousands of genes. Supersecondary structure provides enough information to make reasonable functional annotations, and we can compute it quickly enough to scale to entire genomes. Even threading approaches such as RAPTOR[XLKX03] may require hours per gene. The methods we have developed are faster and more accurate than standard threading approaches.

Threading methods such as RAPTOR [XLKX03] attempt to map a new protein sequence onto templates built from individual solved proteins. While a high-quality threading hit may produce an accurate tertiary structure, this approach loses the ability to take a larger evolutionary view of protein space. Profile-based methods–including SMURF–build knowledge about evolutionarily conserved parts of the protein structure and sequence into their templates. Rather than matching a new protein sequence to a single best-fitting structure, we wish to say that a new sequence belongs to a group of proteins that share evolutionary, structural, and possibly functional similarities.

In order to predict that a new protein sequence shares structure and function with a group of known proteins, we must be sure that these groups of proteins are consistent. In particular, since we use structure to infer function, we wish to ensure that protein space is organized in a structurally consistent way.

1.6 Outline of This Work

In this dissertation, we present several approaches to remedying the complexity of MRFs for remote homology detection, as well as an approach to improving the quality of training data for remote homology detection. Below is the outline of individual chapters in this dissertation.

We begin with a tour of protein fold space (Chapter 2), examining the struc-

tural consistency of the SCOP protein structural hierarchy. We also introduce a method for clustering protein structures such that manually-curated hierarchies such as SCOP [MBH95] can be recreated with reasonable accuracy, based purely on automated structural alignments. We also introduce a benchmark set, called MattBench, that we propose for use by the developers of protein sequence or structural aligners.

In Chapter 3, we discuss an approach to generalizing Markov random fields to the problem of remote homology detection in β-structural proteins. We simplify the SMURF [Men09, MBC10] Markov random field model by limiting the complexity of the dependency graph, in order to bound the computational complexity of finding an optimal parse of a query sequence to a model. We combine these simplified Markov random fields with a model of "simulated evolution" to improve upon existing methods.

In Chapter 4, we introduce an approach for remote homology detection using the SMURF Markov random fields that does not require simplifying the dependency graph. Instead, we introduce a stochastic search approach that quickly computes approximate alignments to the Markov random field, and which should be generalizable to all protein folds.

Finally, in Chapter 5, we discuss the results and summarize the key findings of this dissertation followed by possible directions for future work.

Chapter 2

Touring Protein Space with Matt

2.1 Introduction

Biologists have long relied on manual classification methods to organize the accepted gold-standard hierarchical classification systems for protein structural domains, SCOP [MBH95, AHB+04] and CATH. [OMJ+97, PBB+03, GLA+07] Even now, when both SCOP and CATH have switched to hybrid manual/semi-automated methods [GLA+07], these methods are still attempting to fit new protein domain folds into an initial classification scheme that was derived manually. Expert biologists continue to modify the clustering structure based on sequence, evolutionary, and functional information, not solely based on geometric similarity of the placement of atoms in the protein backbone.

On the other hand, pairwise protein structural alignment programs superimpose protein domains to minimize a distance value based solely on geometric criteria [GL98]. When computational biologists combine such a structural alignment with hierarchical clustering, they obtain a fully automatic, unsupervised partitioning of protein structural domains into hierarchical classification systems [TGG+08]. Such "bottom up" protein structure classifications, as they are called in Valas et

al. [VYB09], have been previously designed based on VAST [MBB95, GMB96], Dali [HS96, HS98, HP00], and others [ZGS+07], and have both practical and theoretical appeal. Practically, researchers can assign new protein structures to clusters more quickly without a human expert. Theoretically, a mathematical characterization of protein similarity and dissimiliarity, if it proves biologically useful or meaningful, is objective, uniformly applied, and gives a human-expert-independent map of the known protein universe.

Unfortunately, multiple researchers have found that SCOP and CATH hierarchical classifications of protein structure both differ substantially from each other [HJ99, GVSD02, BAD03], and also from the classification schemata that result from automatic bottom-up unsupervised clusterings of protein space [GL98, HJ99, SB00, BAD03, STG+06], even when protein chains are broken up into the more modular units of "protein domains," as SCOP, CATH, and most automated schemes now do [HS98, VYB09].

Previous papers have characterized those protein domain clusters on which SCOP and CATH agree [HJ99, GVSD02, BAD03]. Previous automatic methods seem to be able to match the closest-homology *family* level of the SCOP hierarchy, but were found to diverge considerably at the more distantly homologous *superfamily* and at the quite remotely homologous *fold* levels of the SCOP hierarchy [GL98, HJ99, SB00, BAD03, KKL05, STG+06, SWS07], with similar divergence from CATH [HJ99, HPM+02, BAD03]. This is unfortunate, because, for example, the superfamily level of the SCOP hierarchy clusters proteins that share similar topologies and are believed to have evolved from a common ancestor [MBH95], allowing important inferences to be made about function [STG+06, VYB09]. We focus on SCOP rather than CATH for the remainder of this chapter, though much of what we say about SCOP could be applied to CATH. Thus, the superfamily level of the SCOP hierarchy has strong biological utility: if a fully automated, "bottom-up", distance-based clustering method cannot approximately replicate a particular SCOP superfamily, then such a method is not clearly meaningful or useful.

This ties into a spirited debate among the computational proteins community,

about the central question of whether "protein fold space" is *discrete* or *continuous* [Ros02]. A continuous view comes from the theory that modern proteins evolved by aggregating fragments of ancient proteins [Ros02, HPM$^+$02, VYB09, SKG09]. A discrete view comes from evolutionary process constrained by thermodynamic stability of the structure [SKG09]. In particular, if most mutations move the conformation of a stable folded chain away from an "island" of thermodynamic structural stability, then stabilizing selection will promote fold conservation, and movements between folds will be uncommon [CK06]. If geometric distance and evolutionary relation approximately coincide, then an automatic method that approximately matches SCOP at the superfamily level is conceivable.

We present a bottom-up automatic hierarchical classification scheme for protein structural domains based on the multiple structure alignment program Matt [MBC08]. Matt, which stands for "multiple alignment with translations and twists", was specifically developed by our group to geometrically align more distantly homologous protein domains. It accomplishes this by allowing flexibility in the form of small, geometrically impossible bends and breaks in a protein structure, in order to distort that structure into alignment with another protein. Matt was shown to perform particularly well compared to competing multiple and pairwise structure alignment programs on proteins whose homology was similar to the SCOP superfamily level [MBC08, RSWD09, BSL09]. Surprisingly, we find that our automatic classification scheme based on a pairwise distance value derived from Matt, coupled with a straightforward neighbor-joining algorithm to construct the hierarchical clusters [SMP08] matches SCOP better than previous automatic methods, at the superfamily, and even, to some extent, at the fold level. In comparison, the same hierarchical clustering method using a pairwise distance value based on DaliLite [HP00], a recent implementation of the Dali structural alignment algorithm, replicates previous findings and cannot mimic SCOP on the superfamily level of the SCOP hierarchy. We thus conclude that perhaps the threshold at which protein domain space is naturally discrete extends at least through the superfamily level, and that perhaps the manually curated SCOP hierarchy has *geometric* coherence

at the superfamily level (and in some parts of the fold hierarchy, see Section 2.4) so these clusters are intrinsic properties of the geometry of fold space, not just human-generated categories.

A practical implication of our results may be that automatic methods with a Matt-based distance value may ultimately help speed the assignment of new protein structural domains to the appropriate place in the SCOP hierarchy. We note, however, that determining where to place a new structure into an existing hierarchy is a much simpler problem (analogous to "supervised learning") than creating an entire cluster hierarchy from an automatic pairwise distances from scratch (analogous to "unsupervised learning"), and fairly successful methods already exist to correctly place a new structure into the existing SCOP hierarchy [GVSD02, CQKK04, CSX06]. Thus the primary interest in this result may be that if a Matt distance value can "recover" SCOP superfamilies to a great extent, this validates both automatic and hand-curated methods of classification, and the entire concept of "superfamily" at the same time. Namely, at this level of structural similarity, it appears we may not often have to choose between evolutionary and geometric criteria for structural domain similarity.

A byproduct of our organization of protein space is that by looking at where agreement of our Matt clusters with SCOP is exact, we can construct a new set of gold-standard protein multiple structure alignments of distantly homologous proteins (and associated decoy sets) for which we can have confidence that the Matt structural alignment is meaningful. Thus, we introduce "Mattbench," a set of structural alignments at two levels: superfamilies (consisting of 225 alignments with between 3 and 15 proteins in each alignment set), and folds (consisting of 34 alignments with between 3 and 15 proteins in each alignment set). Mattbench is meant as an alternative to the SABmark [VWLW05] benchmark set, which also attempts to mimic SCOP, but Mattbench's alignment sets only cover those subsets of SCOP superfamilies and folds where Matt finds geometric consistency. Thus while Mattbench is slightly less complete than SABmark in coverage, its alignments are likely to be more consistent, making it a better benchmark on which to test sequence alignment

methods. Complete details on how Mattbench is constructed appear in Section 2.2.6; Mattbench itself can be downloaded from `http://www.bcb.tufts.edu/mattbench`.

Finally, we remark that this work, like most recent work that compares different hierarchical classification systems, already presumes the "structural domain" as the basic structural unit (as do SCOP and CATH), where many protein structures contain multiple structural domains [HS98]. The problem of partitioning a protein into its structural domains is far from trivial [VBAS04, HVS06] but there has been much recent progress in computational methods that split a protein structure automatically into domains and find the domain boundaries [HVS06, RHD07]. In any case, that is not the focus of our work, and we assume the protein has already been correctly split into domains as a preprocessing step.

2.2 Methods

2.2.1 Representative Proteins

From the 110,776 protein domains of known structure from ASTRAL version 1.75, we construct a set of representative protein domains filtered to 80% identity (according to BLASTP [AMS$^+$97]) and a minimum sequence length of 40 residues. This provides a reasonable first pass for identifying groups of similar protein domains, and allows us to shrink the search space significantly. The set of clusters is constructed by running a greedy, agglomerative, minimum-linkage clustering algorithm based on this threshold of 80% sequence identity. This produces 10,418 groups of proteins that share significant sequence identity.

From each cluster, we identify a representative. First, we discard engineered or mutant proteins, and any proteins whose X-ray crystallography resolution is > 5.0Å, from any cluster that has alternative representatives that meet our criteria. Next, treating each cluster as a (potentially, but not necessarily, complete) graph whose nodes are the constituent proteins and whose edge weights are the sequence identity values from the BLASTP alignments with at least 80% identity, we consider the weighted degree (sum of edge weights) of each protein, and we favor the proteins

with greatest weighted degree. We break ties first by the date the structure was determined (preferring more recent structures), then by the quality of the solved structure. The remaining ties typically come from sequences with $\geq 99\%$ identity, and we break them arbitrarily. The resulting set has 10,418 representative protein domains.

2.2.2 Distance Values

For these 10,418 representatives, we performed an all-pairs structural alignment using both DaliLite [HP00], the structural aligner used in the FSSP classification scheme [HS98] and Matt [MBC08]. In each case, a distance (or dissimilarity) measure is derived for each pair. For DaliLite, the Z-score proved to be a good measure, so we used it without further modification.

For Matt, we used a new distance value that is a modification of the p-value score computed in Menke, et al. [MBC08]. Let c be the length of the aligned core shared between the two proteins (in residues), r be the RMSD (root mean square deviation) of the alignment, l_1 and l_2 be the lengths of the two protein domains being aligned (in residues), and k_1, k_2, and k_3 be the constants from the Matt p-value. We compute the distance between two Matt-aligned proteins as follows:

$$d = \frac{1}{k_1 \times (r - k_2 \times \frac{c^2}{\frac{l_1+l_2}{2}} + k_3)}$$

This value differs from the formula that Matt uses to compute a p-value only in that it squares the core-length term to place more weight on longer aligned cores (c^2 instead of c). We found this improved performance.

2.2.3 Distance Threshold

Based on each of the Dali Z-score and Matt distances, we next learned the distance cutoffs that most closely mimicked the family, superfamily, and fold levels of the SCOP hierarchy as follows:

In other words, we set $d_{p,q}$ to be the value corresponding to the point on the Receiver Operating Characteristic (ROC) curve that intersects the tangent iso-

Initialize a training set T and a set of already-chosen pairs A;
for $i = 1 \rightarrow 10000$ **do**
  Choose proteins p, q such that $p \neq q$ and p and q are in the same SCOP grouping, and the pair $p, q \notin A$;
  Choose proteins r, s such that $r \neq s$ and r and s are in different SCOP groupings, and the pair $r, s \notin A$;
  $A \leftarrow \{p, q\}$;
  $A \leftarrow \{r, s\}$;
  $T \leftarrow dist(p, q)$ with label *true*;
  $T \leftarrow dist(r, s)$ with label *false*;
  Compute true positive rate R_{tp}, true negative rate R_{tn}, positive rate R_p, and negative rate R_n for T based on the class labels *true* and $false$;
  Determine the value of $d_{p,q}$ that maximizes $\frac{R_{tp}+R_{tn}}{R_p+R_n}$;
end

performance line [VC06], maximizing the sum $R_{tp} + R_{tn}$. The area under the ROC curve measure (AUC) is a summary statistic that captures how well the pairwise distance score can discriminate between structures that share or do not share SCOP cluster membership.

We note that setting the pairwise distance cutoffs (determining the value of $d_{p,q}$ in step 4) is the only "supervision" our algorithm uses in constructing its clustering (see discussion below). We emphasize that once the three single scalar pairwise distance cutoff (corresponding to SCOP 'family', 'superfamily', and 'fold' levels of dissimilarity) are set, *no further information* from SCOP is utilized to produce the clustering.

2.2.4 Clustering and Tree-cutting

Based on the distance functions, we computed values for all pairwise alignments based on the Matt or DaliLite output, and represented this as a distance matrix. We ran the ClearCut program [SMP08] in strict neighbor-joining mode (-N option) to produce a dendrogram based on these Matt or DaliLite distance values. We then recursively descended this tree to produce family, superfamily, and fold-level groupings as follows. For a given subtree, if all leaves (protein domains) in that subtree are within a threshold t of one another (where t is the family, superfamily,

or fold threshold), then those leaves are all merged into a new grouping of that level. Otherwise, we recursively descend into the two subtrees of that subtree's root until we reach a subtree all of whose leaves fall within a given threshold (family, superfamily, or fold; based on Matt distance or DaliLite Z-score as appropriate) of one another. Thus, we are performing a total-linkage clustering, but using the topology of the dendrogram to determine which protein domains get left out of a given cluster.

We remark that Sam et al. [STG+06] did an extensive study of clustering and tree-cutting methods, and looked at their effect on performance for several distance values. They tested 3 "SCOP-dependent" and 7 "SCOP-independent" tree-cutting strategies. However, their "SCOP-independent" strategies all required as input the target number of SCOP clusters to produce at each level. In contrast, our method discovers the number of clusters as an organic function of the protein domain space, based only on a globally learned dissimilarity cutoff; it is thus of independent interest that we nearly replicate the number of SCOP clusters at each level (see Table 2.2).

2.2.5 Jaccard Similarity Metric

The Jaccard index, or Jaccard similarity coefficient, of two sets A and B is defined as $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$. Based on the Jaccard index of a cluster (e.g. family or superfamily or fold) produced by our algorithm (a "Matt family" or "DaliLite family") and a SCOP grouping of the same level, and looking at the identity of protein domains in the two groupings, we can compare how alike they are. We can thus easily find the most similar SCOP family to each Matt family, $S \to M$ and vice versa, $M \to S$. This directional mapping is neither one-to-one nor onto, but each cluster on the 'source' side will be mapped to some most-similar cluster on the 'sink' side. The resulting directed graph allows us to explore the distribution of Jaccard indices as well as the distribution of degrees of each cluster. A perfect matching would correspond to every Jaccard index being 1.0, and every cluster having degree 1. Clearly, we do not expect to achieve a perfect matching, but this metric allows us to compare the quality of clustering, relative to SCOP, of our algorithm using the Matt distance

and the DaliLite Z-score distance.

Each direction of the metric is produced as follows, using as an example the comparison of Matt families to SCOP families. Consider the set of Matt families and SCOP families as a bipartite graph, with the Matt families on one side of the bipartition and the SCOP families on the other. Initially, the graph has no edges. For each Matt family, find the most similar (by Jaccard index) SCOP family. A weighted, directed edge is drawn from each Matt family to its most similar SCOP family; the edge weight is equal to the Jaccard index, which ranges from 0 to 1. This is performed until each Matt family has been matched to a SCOP family. This process is repeated in the other direction, matching each SCOP family to its most similar Matt family, and the same thing is done for Matt and DaliLite at the superfamily and fold levels of the SCOP hierarchy.

Recall that in this analysis, as is standard [HJ99], we are considering only the protein domains that were identified as cluster representatives within each group of protein domains that share 80% sequence identity.

2.2.6 Benchmark Set

Developers of protein sequence aligners–and structural aligners–typically test their alignment quality on gold-standard benchmark sets such as HOMSTRAD [MDBO98] and SABmark [VWLW05]. With the hierarchy of Matt-derived folds, superfamilies, and families constructed, we produced a benchmark set of protein alignments at two levels: superfamilies (consisting of 225 alignments), and folds (also referred to as the "twilight zone" of protein homology, consisting of 34 alignments). The "twilight zone" [Ros99] is the region of low sequence identity (between 20% and 35%) at which homology recognition based upon sequence alignment becomes difficult. At the superfamily level, we generated the benchmark set as follows:

1. Choose Matt superfamilies that contain at least three representative proteins.
2. For each Matt superfamily:
   (a) Identify the most similar SCOP superfamily (by Jaccard index) and take the

intersection of it and the Matt superfamily. Call this set S.

(b) run BLAST on all pairs of proteins in S, storing the maximum e-value as E.

(c) For any pair of proteins $p, q \in S$ that share greater than 50% sequence identity, remove the shorter one (breaking ties arbitrarily by alphabetic order of protein name). Call this set S'. Proceed if and only if S' still has at least three proteins.

(d) Run a Matt multiple alignment on S', and store this alignment as the Mattbench alignment for S'

3. For each Mattbench superfamily S, produce a decoy set D as follows:

(a) Consider every Matt representative protein $p \notin S$. For each p:

i. discard p if it is in the most similar (by Jaccard index) SCOP superfamily to p's Matt superfamily

ii. run BLAST on p against every protein $s \in S$, storing the e-value $e_{s,p}$ and sequence identity $i_{s,p}$

iii. run Matt on p against every protein $s \in S$, storing the Matt distance $m_{s,p}$

iv. discard p if $\exists s$ such that $i_{s,p} \geq 50\%$

v. discard p unless $\exists s$ such that $e_{s,p} < E$ (this is the E stored as the maximum e-value above)

vi. discard p unless $\forall s, m_{s,p} > T_{superfamily}$ (the superfamily threshold used in Matt clustering)

vii. if p has not been discarded, add it to the benchmark decoy set D.

The "twilight zone" benchmark set is generated in an identical manner, except that the Matt and SCOP fold levels are used, and the sequence identity cutoff is 20% rather than 50%. The BLAST E-value criterion is the same used by SABmark [VWLW05] and ensures that each decoy is a useful decoy rather than an obvious negative match. The Matt distance criterion is present because, if the decoy protein is within the threshold of some protein in that superfamily, the decoy is only *not* in that superfamily because of the overall topology of the cluster–that is, because while the decoy may be similar to some protein in that cluster, it is not similar enough to all of the proteins of that cluster to warrant inclusion. The purpose of the decoy set is to act as a set of likely false positives, that a sequence

aligner will find challenging to distinguish from the true positives. Both benchmarks can be found at `http://www.bcb.tufts.edu/mattbench`.

2.3 Results

2.3.1 Pairwise Distance Comparisons

Table 2.1: ROC Area for pairwise performance vs. SCOP

	Matt	DaliLite
Families	0.922	0.958
Superfamilies	0.842	0.615
Folds	0.840	0.871

Note: While DaliLite slightly outperforms Matt at family and fold levels, Matt significantly outperforms DaliLite at the superfamily level.

We first asked if a pairwise Matt or DaliLite distance cutoff could correctly distinguish among pairs of proteins that were in the same SCOP cluster from those that were not. Table 2.1 shows the ROC area at the SCOP family, superfamily, and fold level, for the Matt and DaliLite distance scores. Note that at the family and fold levels, these values are very close (DaliLite outperforms Matt by a small margin), but at the superfamily level, Matt significantly outperforms DaliLite, achieving 0.842 ROC area vs. DaliLite's 0.615. Matt was developed to better align structures at the superfamily level of homology, but the size of the gap in ROC area is still surprising. We further remark that at the fold level, DaliLite's seemingly competitive performance is somewhat illusory, since it is shattering many SCOP folds, each into many tiny pieces (see below).

2.3.2 Clustering Performance

While the pairwise performance of Matt compared to DaliLite at the superfamily level is impressive, pairwise similarity does not necessarily translate into better clustering performance. Thus, we next explore Matt's clustering performance.

Table 2.2: Number of clusters at each level for each method

	SCOP	Matt	DaliLite
Families	3471	3498	3081
Superfamilies	1656	1716	2455
Folds	981	891	2277

Note: Matt more closely matches the number of families, superfamilies, and folds in SCOP than DaliLite does. DaliLite clustering results in too few families, but too many superfamilies and folds with respect to SCOP.

First we give the simplest possible comparison: raw numbers of clusters produced by Matt and DaliLite compared to SCOP at the three levels. Recall that unlike the clustering algorithm explored by Tai, et al. [TGG+08], the number of clusters produced by our dendrogram and tree-cutting method is a direct consequence of the pairwise distance threshold, and is not artificially set to match SCOP (see Section 2.2.4). Table 2.2 shows that the Matt clustering produces approximately the same number of clusters as SCOP at all three levels. While DaliLite also produces approximately the same number of clusters at the family level, at the superfamily and fold levels it produces many more clusters than SCOP. We next explore exactly how both methods split and merge SCOP clusters in more detail.

The Jaccard index serves as a good indicator of how well Matt and DaliLite match SCOP. As the raw numbers of clusters in Table 2.2 suggest, DaliLite often shatters SCOP superfamilies into multiple clusters. DaliLite also shatters SCOP folds into many more shards on average than Matt. How can this be given the very similar pairwise classification performance at the fold level? We defer this question until Section 2.4. We note that even at the family level, Matt performs slightly better than DaliLite at both the average degree and average Jaccard similarity metrics. The average number of Matt or DaliLite families that match to a single SCOP family is between 3.5 and 4; however, notice that a large majority of Matt or DaliLite families map to a single SCOP family and the average is pulled up by a few outliers (see histograms in Figures 2.1 and 2.2). Average degree values at the superfamily and fold levels stay nearly constant for Matt, whereas DaliLite's average

Table 2.3: Descriptive statistics for the family, superfamily, and fold levels

Family	Max Deg.	μ Deg.	σ Deg.	Min Sim.	μ Sim.	σ Sim.
Matt →SCOP	30	3.63	5.470	0.005	0.611	0.373
DaliLite → SCOP	45	3.902	6.919	0.001	0.598	0.380
SCOP → Matt	15	1.873	2.160	0.127	0.712	0.336
SCOP → DaliLite	12	1.983	1.823	0.001	0.655	0.347
Superfamily	Max Deg.	μ Deg.	σ Deg.	Min Sim.	μ Sim.	σ Sim.
Matt → SCOP	28	3.633	5.094	0.003	0.587	0.389
DaliLite → SCOP	153	16.61	36.54	0.001	0.428	0.406
SCOP → Matt	15	1.704	1.913	0.020	0.714	0.326
SCOP → DaliLite	10	1.470	1.229	0.001	0.713	0.324
Fold	Max Deg.	μ Deg.	σ Deg.	Min Sim.	μ Sim.	σ Sim.
Matt → SCOP	18	3.719	4.258	0.004	0.467	0.354
DaliLite → SCOP	149	26.57	40.87	0.001	0.321	0.389
SCOP → Matt	6	1.958	1.122	0.022	0.512	0.326
SCOP → DaliLite	3	1.117	0.353	0.001	0.758	0.299

Note: μ Degree is the average number of clusters from the first scheme that map to a single cluster in the second, and σ Degree gives the standard deviation. Similarly, we give min, μ, and σ of the Jaccard similarity.

degree values rise to 16.61 for the superfamily level and 26.57 at the fold level. In the other direction, considering how many Matt or DaliLite clusters span multiple SCOP clusters, at the family level the average degree for Matt and DaliLite are nearly identical (between 1.8 and 2). At the superfamily and fold levels, we would expect DaliLite to outperform Matt by virtue of the fact that it creates many smaller clusters (see Table 2.2), and DaliLite does, but by a fairly small margin (1.4 to 1.7 at the superfamily level and 1.1 to 2 at the fold level). The distributions are displayed in more detail in the histograms in Figures 2.1,2.2, 2.3, 2.4, 2.5, and 2.6.

2.3.3 Specific Example

We thought it would be illuminating to provide a pictorial example of a single SCOP superfamily that Matt splits into two superfamilies. Consider the SCOP superfamily "DHS-like NAD/FAD-binding domain" (SCOP ID 52467). There are 24 proteins from this superfamily in our representative set. Matt places 17 of them in one superfamily, but the remaining 7 in a different superfamily. Figure 2.7a gives

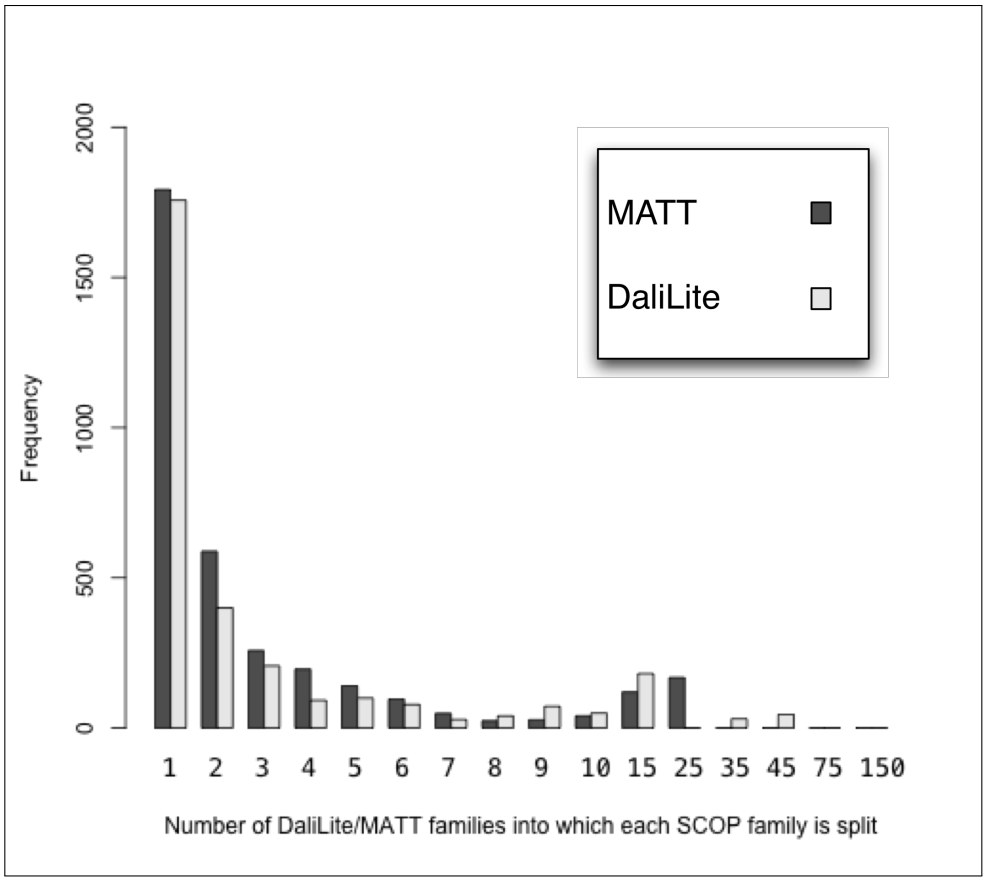


Figure 2.1: Number of Matt vs. DaliLite families into which each SCOP family is shattered.

an example protein from the Matt superfamily of size 17, while Figure 2.7b gives an example protein from the Matt superfamily of size 7. Both Matt superfamilies contain the same single flat β-sheet of 6 or 7 strands, surrounded by α-helices. In addition, the proteins in the Matt superfamily of size 7 have a second short 3-4 strand β-sheet. The second short β-sheet is physically on one end of the first β-sheet in 3-dimensional space, but sometimes occurs between the second-to-last and last β-strands in the first β-sheet in terms of linear (sequence) ordering, or else at the very end. The second β-strand is also partially surrounded by α-helices.

Because of the common central motif, it is very possible that these proteins are evolutionarily related and thus belong in the same SCOP superfamily. However, geometrically, the additional short β-sheet is significant enough for Matt to place them in different superfamilies. Matt does, however, place them in the same fold.

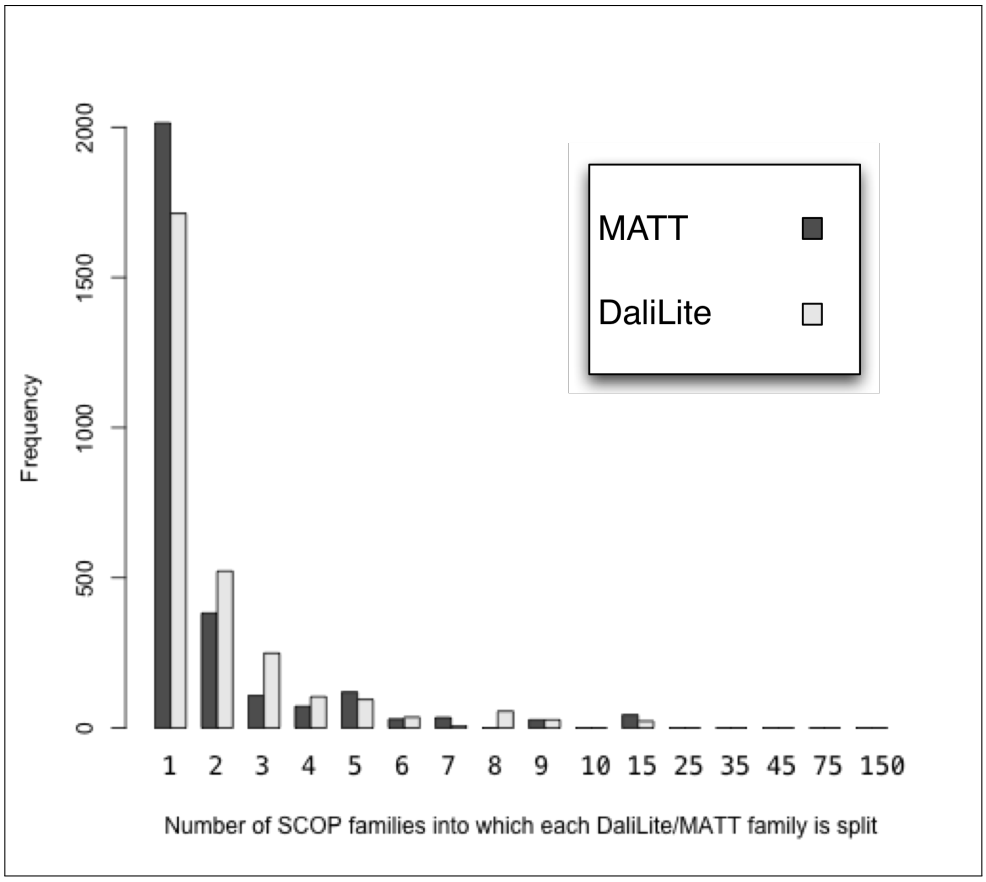


Figure 2.2: Number of SCOP families into which each Matt or DaliLite family is shattered.

2.4 Discussion

We have shown that using more modern structure alignment programs, we can approximately match SCOP at the superfamily level. Of course, any mapping between one set of clusters based on geometric equivalence, and another set of clusters based on geometric as well as evolutionary equivalence, will be imperfect–yet the Matt clusters at the superfamily level seem sufficiently interesting that differences between Matt and SCOP could be illuminating.

As noted earlier, DaliLite tends to shatter SCOP folds into many more shards than Matt. How can this be, given the very similar pairwise classification performance at this level? One possibility is that the Matt-based distance value is more stable in regions far beyond the specific thresholds we learned, and that this leads to

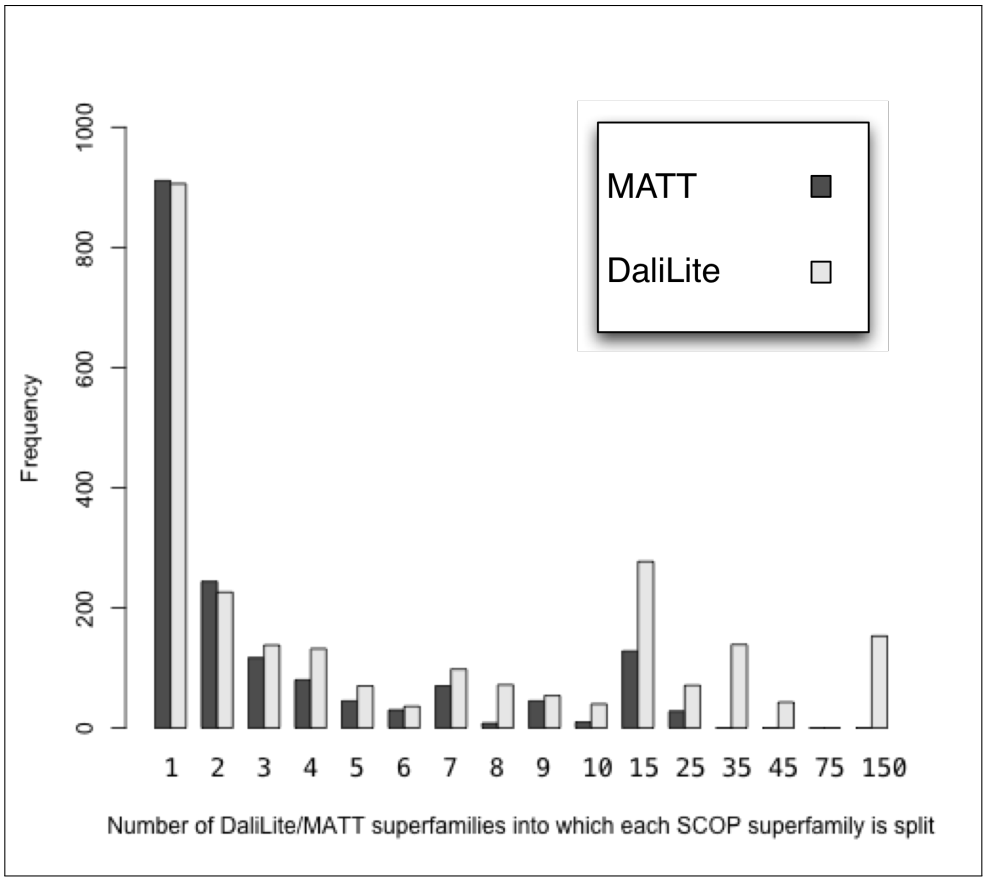


Figure 2.3: Number of Matt vs. DaliLite superfamilies into which each SCOP superfamily is shattered. Note the tail of the distribution, in which DaliLite breaks SCOP superfamilies into many small pieces.

the topology of the resulting dendrogram (before cutting) more faithfully representing the relationships between more- and less-closely related folds. In other words, DaliLite's Z-scores may result in more 'spoilers'–individual proteins with large distances to many other proteins in the same cluster–that break up clusters (due to our total-linkage requirement) than Matt's distance value. While we have only compared Matt to DaliLite, comparisons to other aligners such as TM-Align [ZS05] would undoubtedly be interesting. We focused on the comparison to DaliLite due to it being the aligner underlying the FSSP database [HS98].

What do Matt's clustering results mean for protein fold space at the "fold" level of structural homology? Here, while the Matt clustering clearly seems more informative than that produced by DaliLite, performance is still uneven. There seem

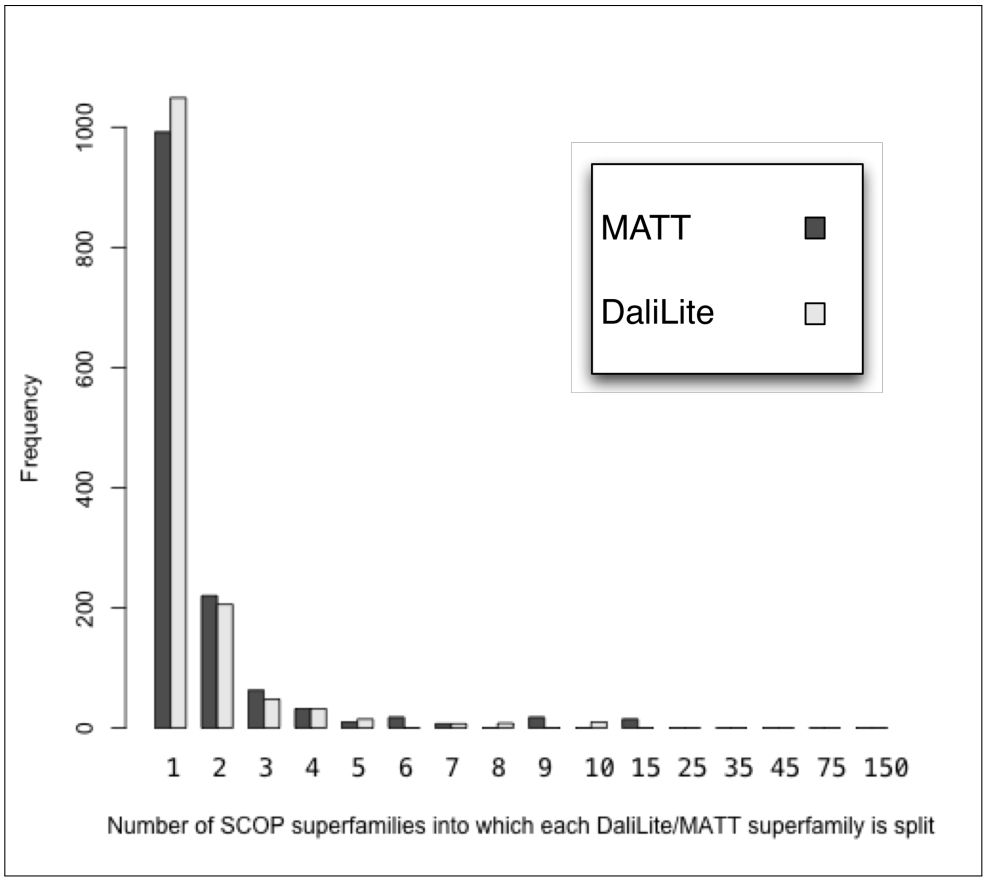


Figure 2.4: Number of SCOP superfamilies into which each Matt or DaliLite superfamily is shattered.

to be some SCOP folds where the Matt split appears meaningful, and others where it is more arbitrary. For example, a notoriously difficult SCOP fold for multiple automatic methods is the enormous β/α TIM barrel fold. SCOP places 33 separate superfamilies into this one fold, but both of our clustering approaches seem to split it into multiple folds. For example, DaliLite splits the TIM barrel SCOP fold into 106 separate folds. Matt splits the TIM barrel SCOP fold into 'only' 17 separate folds, which is better than 106, but inspection of the boundaries between these Matt fold classes shows more continuity of shape, and the cuts appear to be somewhat arbitrary.

Thus, while touring protein space with Matt seems to lend support to a more discrete view of protein space through the superfamily level, further study of individual clusters may be warranted to determine the breakpoint distance at which

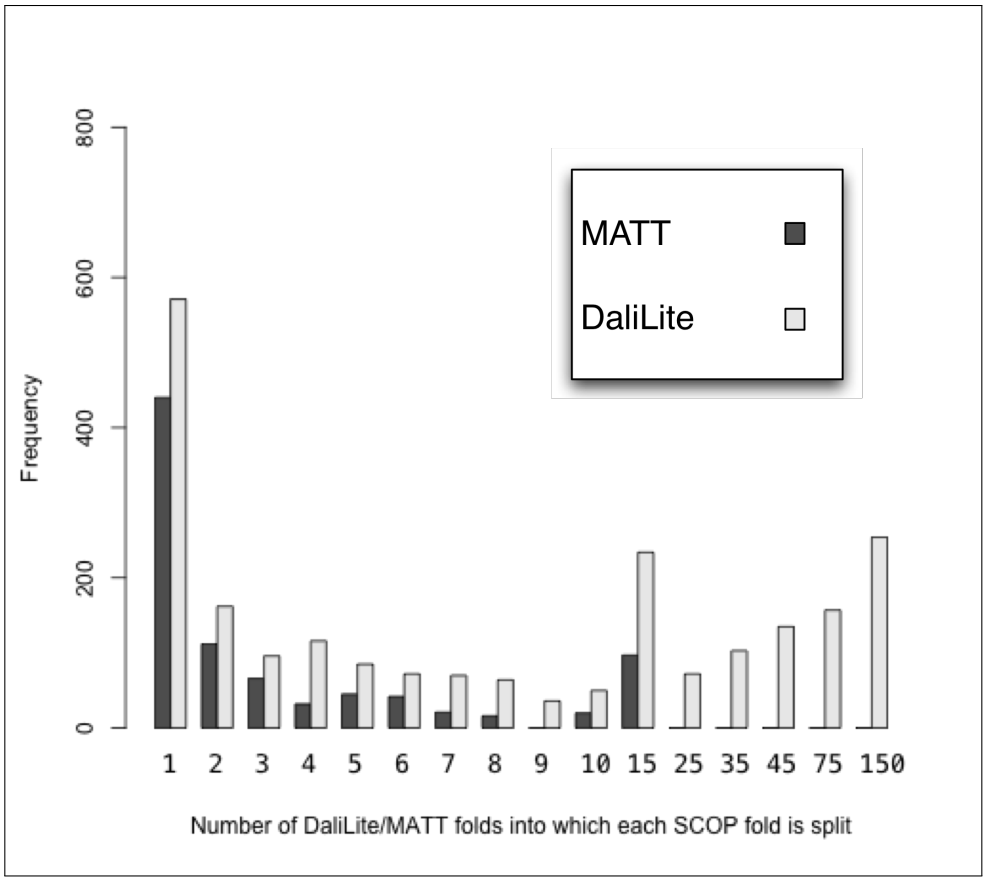


Figure 2.5: Number of Matt vs. DaliLite folds into which each SCOP fold is shattered. Note the tail of the distribution, in which DaliLite breaks SCOP folds into many small pieces.

continuity takes over. Perhaps the degree of similarity of different individual SCOP folds can be characterized, similarly to what Suhrer, et al. [SWS07] did at the family level.

We have made the Mattbench benchmark set available at `http://www.bcb.tufts.edu/mattbench`. We hope that developers of protein sequence alignment tools will consider testing their performance on Mattbench, as well as SABmark [VWLW05] and HOMSTRAD [MDBO98].

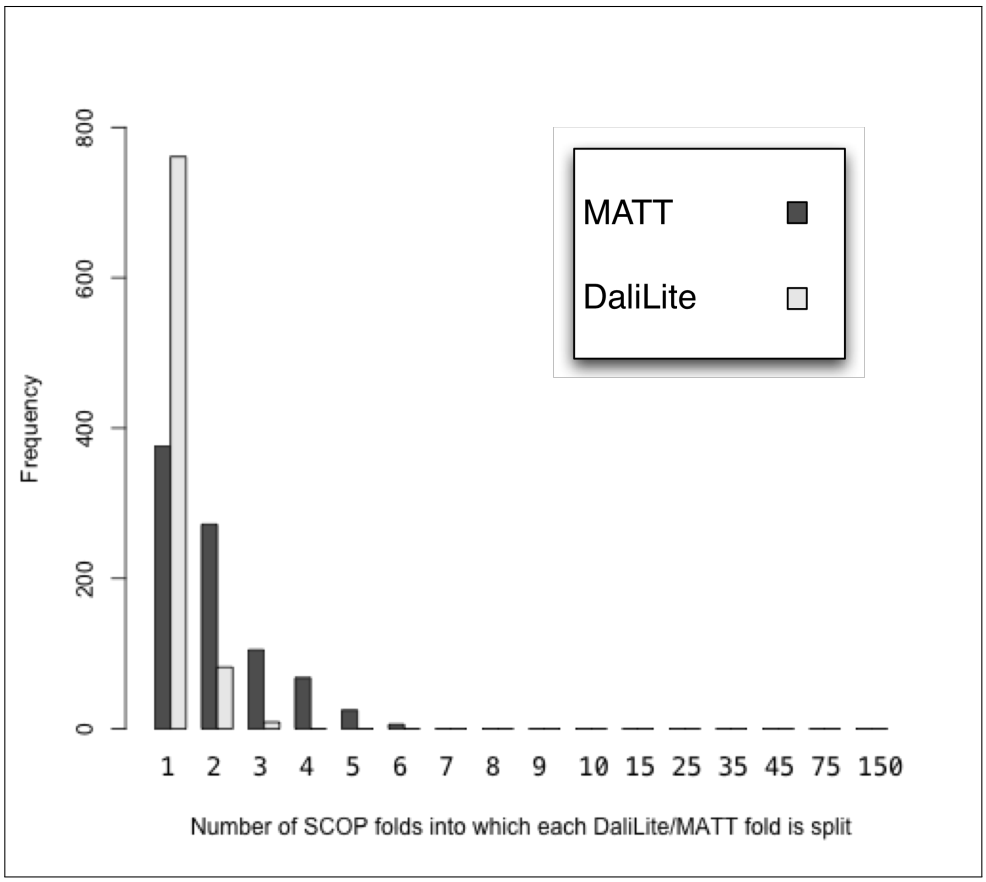
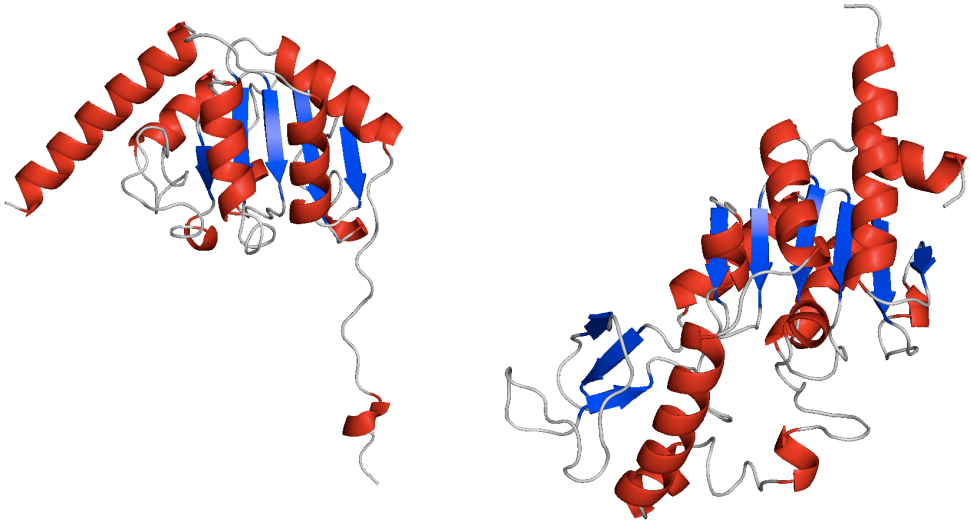


Figure 2.6: Number of SCOP folds into which each Matt or DaliLite fold is shattered.



(a) **Example from Matt superfamily 252** Pyruvate oxidase from *Lactobacillus plantarum*, PDB ID 2ez9:a

(b) **Example from Matt superfamily 212** AF1676, Sir2 homolog (Sir2-AF1) from Archaeon *Archaeoglobus fulgidus* , PDB ID 1m2k:a

Figure 2.7: Example of a SCOP superfamily split by Matt

Chapter 3

Simplified Markov Random Fields and Simulated Evolution Improve Remote Homology Detection for Beta-structural Proteins

3.1 Introduction

Many researchers use hidden Markov models (HMMs) to annotate proteins according to homology, with popular systems such as Pfam ([FTM$^+$08]) and Superfamily ([WMV$^+$07]) based on HMM methods integrated into UniProt. However, HMMs are limited in their power to recognize remote homologs because of their inability to model statistical dependencies between amino-acid residues that are close in space but far apart in sequence ([LS80, ZB99, ORV99, CBM$^+$02, ST02]).

For this reason, many have suggested ([WMS94, LS96, TRBK08, GW09, MBC10, PX11a]) that more powerful Markov random fields (MRFs) be used. MRFs employ an auxiliary *dependency graph* which allows them to model more com-

plex statistical dependencies, including statistical dependencies that occur between amino-acid residues that are hydrogen-bonded in β-sheets.

However, as the dependency graph becomes more complex, major design difficulties emerge. First, the MRF becomes more difficult to train. Second, finding the optimal-scoring parse of the target to the model quickly becomes computationally intractable.

We have built a fully automated system, SMURFLite, that combines the power of Markov random fields with Kumar and Cowen's Simulated Evolution ([KC10]) (which offloads information about pairwise dependencies in β-sheets into new, artificial training data), in order to build the first MRF models that are computationally tractable for *all* β-structural proteins, even those with limited training data. The SMURFLite system builds in part on the SMURF MRF ([MBC10]), which uses multidimensional dynamic programming to simultaneously capture both standard HMM models and the pairwise interactions between amino acid residues bonded together in β-sheets.

Unlike the full SMURF MRF, where the computational requirements of the random field become prohibitive on folds with deeply interleaved β-strand pairs, such as barrels, SMURFLite is tractable on all β-structural proteins (see Figure 3.3). SMURFLite enables researchers to trade modeling power for computational cost by tuning an *interleave threshold*. The interleave threshold represents the maximum number of unrelated β-strands that can occur in linear sequence between the β-strands hydrogen-bonded in a β-sheet while still being retained as pairwise dependencies in the MRF. As the interleave threshold increases, computation time increases, but so does the power of the MRF (see Figure 3.2).

We first test SMURFLite on all propeller and barrel folds in the mainly-β class of the SCOP hierarchy in stringent cross-validation experiments. We show a mean 26% (median 16%) improvement in Area Under Curve (AUC) for β-structural motif recognition as compared to HMMER ([Edd98]) (a popular HMM method) and a mean 33% (median 19%) improvement as compared to RAPTOR ([XLKX03]) (a well-known threading method), and even a mean 18% (median 10%) improvement in

AUC over HHPred ([Söd05, SBL05]) (a profile-profile HMM method), despite HHpred's use of extensive additional training data. We demonstrate SMURFLite's ability to scale to whole genomes by running a SMURFLite library of 207 β-structural SCOP superfamilies against the entire genome of *Thermotoga maritima*, and make over a hundred new fold predictions (available at `http://smurf.cs.tufts.edu/smurflite`). The majority of these predictions are for genes that display very little sequence similarity with any proteins of known structure, demonstrating the power of SMURFlite to recognize remote homologs.

We offer an online server (`http://smurf.cs.tufts.edu/smurflite`) for predicting remote homologs from our library of 207 mainly-β superfamilies using SMURFLite. The online server sets the interleave threshold (the parameter that determines the complexity of the MRF) to 2; we have also shown that increasing the interleave number for SMURFLite can dramatically improve accuracy, but at a great computational cost. While the primary intent of using simulated evolution in conjunction with simplified MRFs is to compensate for the removal of highly-interleaved β-strand pairs required for computational feasibility, we surprisingly find that simulated evolution can still improve full-fledged SMURF in cases of sparse training data. For instance, the 5-bladed β-propellers have only three superfamilies in SCOP, two of which contain only one family. We find that for the 5-bladed β-propeller fold, combining SMURF and simulated evolution improves AUC from 0.73 for full SMURF alone to 0.89.

3.2 Methods

3.2.1 Summary of SMURF Markov random field framework

SMURF and SMURFLite rely on training data in the form of a multiple structure alignment with β-strand annotation. This alignment is created using the Matt program ([MBC08]). β-strand annotation is done on a structure-by-structure basis, where the β-strand residue pairing is determined using the same algorithm implemented by the Rasmol ([SMW95]) visualization program. Essentially, β-strands are

detected by analyzing the ψ, ϕ, and ω angles, as well as the distance between a hydrogen from the amine group and an oxygen from the carboxyl group of the amino acid at which that hydrogen is pointing, if any. If this hydrogen and oxygen point at each other and are within 3Å, they are considered to be hydrogen-bonded. If successive hydrogen bonds are to amino acids near (3-5 residues) in sequence, an α-helix is inferred; if those bonds are to distant amino acids in sequence, a β-strand is inferred. A postprocessing step annotates those β-strand residues that appear in more than half the structures in the alignment as β-conserved. As gaps in β-strands would complicate training, this post-processing step makes β-conserved template strands contiguous in the alignment exactly as in [MBC10]. Specifically, any gaps in a column, that otherwise comprises at least half β-structural amino acids, are removed from the alignment. Recall that up to half the sequences are allowed to *not* participate in β-strands at any given position of the alignment, the non-β-strand amino acids in those positions are still treated as if they participate in β-strands.

The result at this stage is a sequence alignment (resulting from the Matt structural alignment) with conserved β-strand pairs annotated according to the residue positions and conformation (buried or exposed to solvent).

The pairwise probability portion of the MRF is based on the β probability tables that were computed by collecting a set of amphipathic β-sheets from the Protein Data Bank (PDB) ([BBB$^+$00]) and tabulating the frequencies of pairs of hydrogen-bonded residues in two tables, one for buried residues and one for residues exposed to solvent ([BCM$^+$01], [MBC10]). The β-structural proteins chosen were filtered to 25% sequence identity to prevent over-representation of highly-sampled sequences. Amphipathic β-sheets are those β-sheets that are "confused" as to their hydrophobicity, and thus have residues whose sidechains may alternate as to the direction in which they pack. For each residue position, the most likely conformation (buried or exposed) is chosen based on whether that residue pairing is most probable from the buried or exposed β-pairing tables.

Given a trained MRF, SMURF and SMURFLite align a query sequence to the MRF. The query phase of SMURF and SMURFLite computes the alignment of

the sequence to the states of the MRF that maximizes the combined score:

$$log\,(\text{HMM score}) + log\,(\text{pairwise score})$$

In this combined score, the HMM score is the conditional probability of observing the sequence given the HMM portion of the model, and the pairwise score is the conditional probability of observing the paired β-strand components of the sequence given the β-pair portion of the model. Let the sequence have residues $r_1..r_n$, and the MRF have match states $m_1..m_l$, deletion states $d_1..d_l$, and insertion states $i_1..i_l$. Suppose that $r_1..r_k$ and match states $m_1..m_s$ have been assigned. Then, the probability of assigning r_k to the next match state $m_j = m_{s+1}$ is:

$$\begin{aligned} Pr\,[m_j|r_k, u_{j-1}] = HMM\,[m_j, r_k] \cdot \\ transition\,[u_{j-1}, m_j] \cdot \\ \beta strand\,[r_j, r_k, m_j, m_k] \end{aligned}$$

where u_{j-1} can be either d_{j-1}, i_{j-1}, or m_{j-1} depending on whether the current state is a deletion, insertion, or match state. When the current state is a match state, the SMURFLite template replaces the $transition\,[u_{j-1}, m_j]$ term with a value of 1. The $\beta strand$ component is set to be identically 1 unless the particular match state m_j participates in a β-strand that is matched with a state m_k earlier in the template. This component is the primary difference between our MRF and an ordinary HMM ([MBC10]).

SMURFLite computes the maximum score of a sequence using multidimensional dynamic programming on the MRF. This dynamic programming resembles the classic Viterbi algorithm ([Vit67]) used on HMMER's "Plan7" ([Edd98]) HMMs, except that some states are β-strand states, which are required to be match states, and which are paired with other β-strand nodes in the model. Because the pairwise component of the score can only be calculated for a given MRF node once it is determined what residue occupies the paired MRF node earlier in the sequence, each

time the dynamic programming reaches a state in the MRF that corresponds to the first residue of the first β-strand in a set of paired β-strands, we need to keep track of multiple cases, depending on what residue in sequence is mapped to that state. SMURFLite uses a multidimensional array to memoize these possible subproblem solutions. A maximum gap size is set to the longest gap seen in the training data plus 20, for computational efficiency. When paired β-strands follow each other in sequence with no interleaving β-strands between them, the number of dimensions in the table for the dynamic programming is directly proportional to the maximum gap length. Let us call the last MRF state for the first of every pair of β-strands a "split" state and the first MRF state for the second of that pair a "join" state. Then, at every split state, the number of dimensions of the dynamic program will be multiplied by the maximum gap length, because the dynamic program must keep track of scores for each possible sequence position (up to the maximum gap length) that could be mapped to that state. At the corresponding join state, the number of dimensions will be reduced by the maximum gap length, because the scoring function can calculate all the pairwise probabilities of placing that residue into the join state, and then simply take the maximum of all ways to have placed its paired residue into the split state. However, when other β-strands are interleaved, the dynamic program must open additional multidimensional tables before clearing the previous ones from memory. An example of this interleaving is shown in Figure 3.3. Thus, the number of elements in the multidimensional table is never more than the sequence length times the maximum gap length raised to the power of the interleave number.

3.2.2 Datasets

From SCOP ([MBH95]) version 1.75, we chose the folds "5-bladed Beta-Propellers", "6-bladed Beta-Propellers", "7-bladed Beta-Propellers", and "8-bladed Beta-Propellers". We also chose superfamilies from all of the mostly-β folds containing the word "barrel" in their description, whether open or closed, restricted to those superfamilies comprising at least four families (in order to facilitate leave-family-out cross-

validation). These superfamilies were: "Nucleic acid-binding proteins" (50249), "Translation proteins" (50447), "Trypsin-like serine proteases" (50494), "Barwin-like endoglucanases" (50685), "Cyclophilin-like" (50891), "Sm-like ribonucleoproteins" (50182), "PDZ domain-like" (50156), "Prokaryotic SH3-related domain" (82057), "Tudor/PWWP/MBT" (63748), "Electron Transport accessory proteins" (50090), "Translation proteins SH3-like domain" (50104), "Lipocalins" (50814) and "FMN-binding split barrel" (50475). Of these, we removed the superfamilies "Lipocalins" and "Trypsin-like serine proteases," which were not structurally consistent enough to permit a multiple structure alignment for training HMMER or the SMURF variants, and which were broken into distinct superfamilies by [DKCM11], with the result that 11 superfamilies containing barrels were selected. In addition, for the whole-genome search on *Thermotoga maritima*, out of 354 total superfamilies within the SCOP class "All beta proteins", 288 (81%) of which contain at least two protein chains, 207 superfamilies (71%) were structurally consistent enough to be aligned using the Matt ([MBC08]) structural alignment program. We built SMURFLite templates for these 207 superfamilies, and obtained from UniProt the protein sequences for *Thermotoga maritima*, comprising 1852 genes.

3.2.3 Training and testing process

For the β-propeller folds, strict leave-superfamily-out cross-validation was performed. The propeller folds are structurally highly consistent ([MBC10]), and thus high-quality multiple structure alignments were possible using Matt ([MBC08]) without descending to the superfamily level. For each propeller fold, its constituent superfamilies were identified. For each superfamily, one pass of cross-validation was performed. Given a superfamily to be left out, a training set was established from the protein chains in the remaining superfamilies, with duplicate sequences removed. An HMM (in the case of HMMER and HHPred) or MRF (in the case of SMURF and SMURFLite) was trained on the training set (HMMER parameter settings are discussed below). Protein chains from the left-out superfamily were used as positive test examples. Negative test examples were protein chains from all other folds in

SCOP classes 1, 2, 3 and 4 (including propeller folds with differing blade counts), indicated as representatives from the non-redundant Protein Data Bank repository (nr-PDB) ([BBB+00]) database with non-redundancy set to a BLAST E-value of 10^{-7}.

The β-propellers are atypical of most β-structural SCOP folds, in that they structurally align well at the fold level of the SCOP hierarchy. For the β-barrel superfamilies, strict leave-family-out cross-validation was performed. The barrel superfamilies are distinguished by strand number and shear as well as other structural features ([MBH95]), and so like most β-structural motifs they do not align well structurally at the fold level. For this reason, the superfamily level was chosen for training. This cross-validation was similar to that chosen for the β-propellers, except that it was done at the superfamily level, and thus each pass of the cross-validation involved leaving out a family and training on a structural alignment of representatives from the remaining families in that superfamily.

Each test example was aligned to the trained HMM (from HMMER and HHPred) and MRF, and was also threaded, using RAPTOR, against each individual chain in the training set (RAPTOR parameters are discussed below). The score reported for HMMER and HHPred was the output HMM score, and the score reported for SMURF and SMURFLite was the combined HMM and pairwise score from the MRF. For RAPTOR, the score reported for a test example was the highest score from all the scores resulting from threading that test example onto each chain in the training set. For each training set, the scores for each method were collected and a ROC curve (a plot of true positive rate versus false positive rate) computed. We report the area under the curve (AUC statistic) from this ROC curve ([SKP08]).

3.2.4 p-values

SMURFLite computes the p-value for an alignment in a similar manner to HMMER, using an extreme value distribution (EVD) ([Edd98]). An EVD is fitted to the distribution of raw scores over a random sampling of 5000 protein chains from across the SCOP hierarchy. The p-value is then simply computed as $1 - cdf\,(x)$ for any

raw SMURFLite score x, where *cdf* is the cumulative distribution function for the EVD.

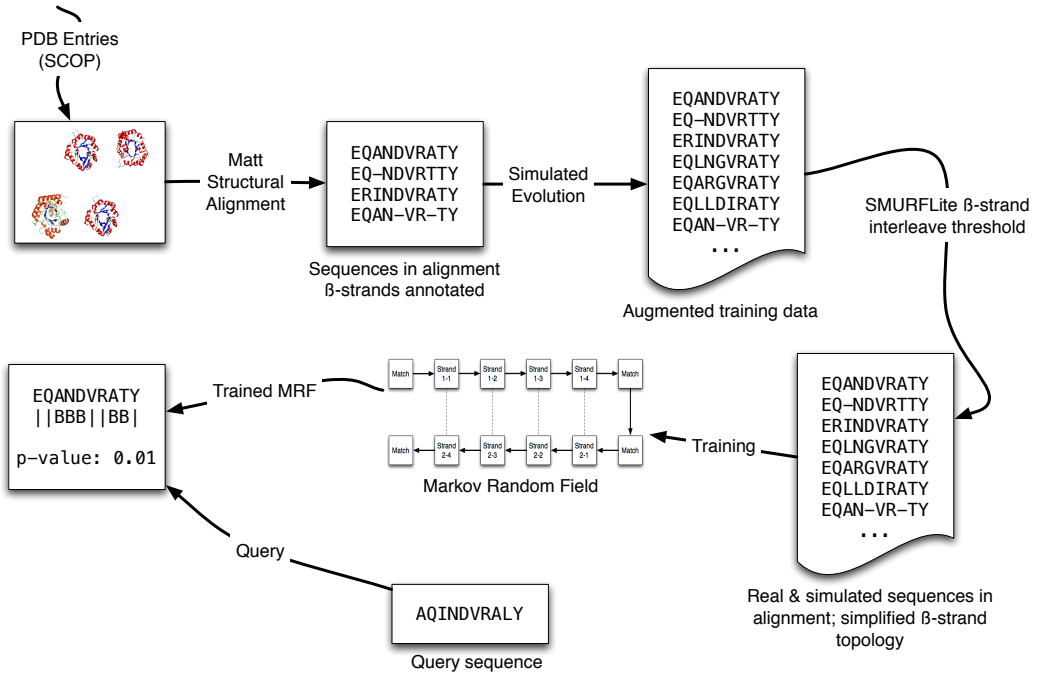


Figure 3.1: The SMURFLite pipeline, including simulated evolution and simplification of the β-strand topology

3.2.5 SMURFLite augmented training data

Kumar and Cowen [KC09, KC10] showed that "simulated evolution," augmenting limited training data with additional sequences produced by mutating the original sequences, improved the performance of HMMER at recognizing the same-superfamily level of homology. Kumar and Cowen [KC10] used two types of simulated evolution: point-wise and pairwise. Here we add only pairwise mutations based on β-strand pairings, as we expect long-range interactions between β-strands to be highly conserved across similar structures. We postulated that the elimination of the β-strand pairs SMURFLite must disregard because of computational complexity might be compensated for by augmenting the training data with artificial sequences based on likely mutations in those paired β-strands. This training-data augmentation comes at insignificant runtime cost and is done before β-strand pairs are

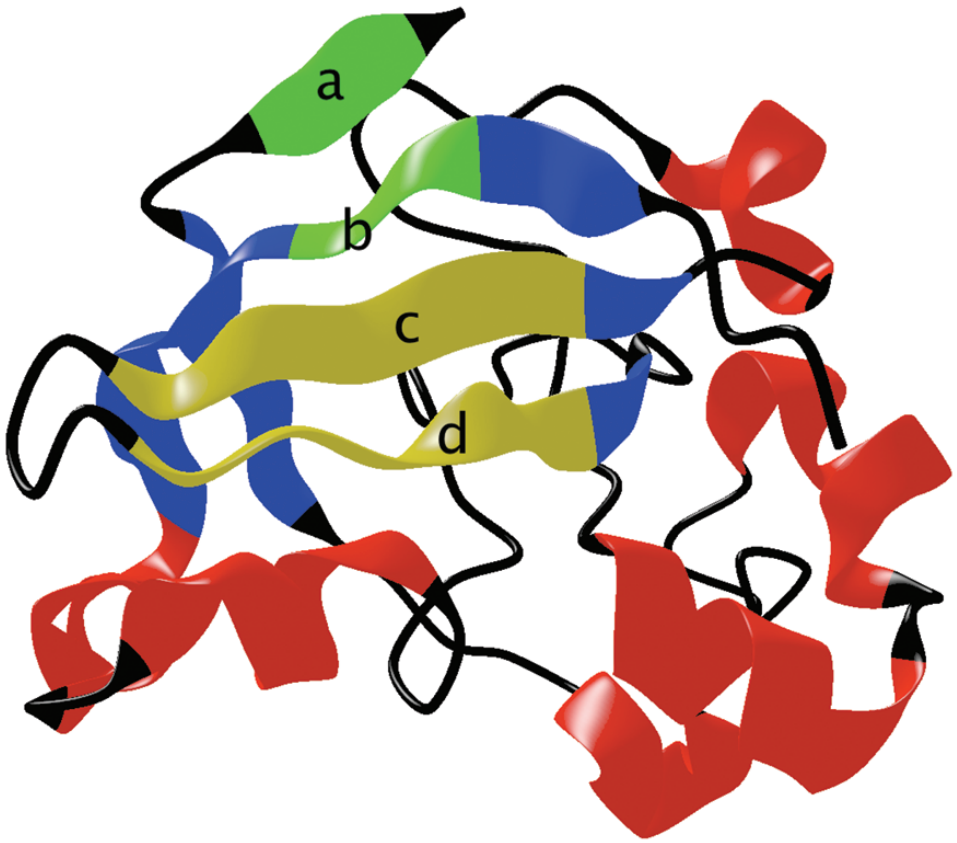
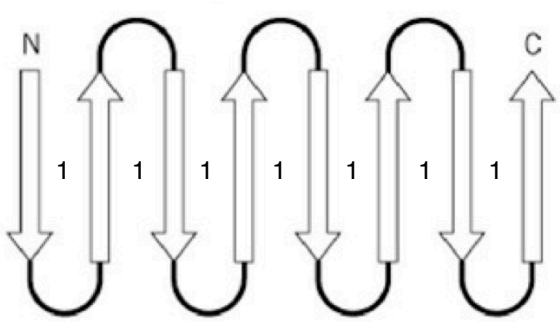
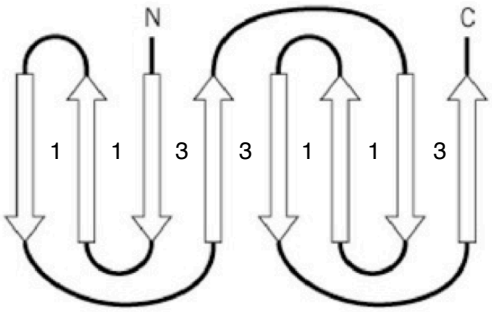


Figure 3.2: A closed β-barrel (PDB ID 1bw3, a Barwin domain) from the superfamily "Barwin-like endoglucanases" to illustrate interleaving of strand pairs. β-strands a and b, which close the barrel, have interleave 4, while strands c and d, which are adjacent in sequence, have interleave 1. Strands b and c have interleave 2. This is because, if we begin at the N-terminal end, the order of the β-strands is a, c, d, b.

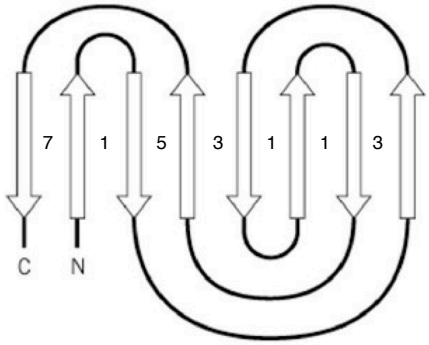
removed from the template (but after their interleave number has been identified, where we define interleave number next below). The mutation frequencies are given by the Betawrap and SMURF ([BCM$^+$01, MBC10]) pairwise probability tables. Using the same algorithm as [KC10], we generate 150 new artificial training sequences from each original training sequence. For each artificial sequence, we mutate at a 50% mutation rate per length of the β-strands. The parameters 150 and 50% were recommended by [KC10]; we also evaluated a 10% mutation rate (a secondary peak according to their work) and performance was slightly worse (data available from the authors).



(a)



(b)



(c)

Figure 3.3: **(a)** An "up-and-down" β-sheet. Unless the C-terminal and N-terminal ends are hydrogen-bonded together, the interleave is 1 for each pair of strands.
(b) A "greek key" β-sheet. The numbers between each pair of β-strands indicate the interleave. The maximum interleave in this instance is 3.
(c) A "jelly roll" β-sheet. The numbers between each pair of β-strands indicate the interleave. The maximum interleave in this instance is 7, between the C-terminal and N-terminal strands.

3.2.6 SMURFLite simplified random field

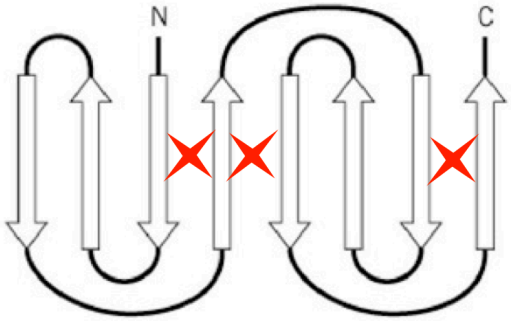
SMURFLite trains a MRF on a template built from a multiple structure alignment. β-strands in the aligned set of structures are found by the program SmurfPreparse which is part of the SMURF package ([Men09, MBC10]). The program not only outputs the positions of the consensus β-strands in the alignment, it also declares a position buried or exposed based on which of the two tables is the best fit to the amino acids that appear in that position in the training data. SMURFLite then assigns an interleave value to each β-strand pair, as follows: Any pairwise interaction between β-strands whose interleave value equals or exceeds the SMURFLite threshold is removed from the training data.

Consider three β-strands: A, B, and C. Suppose strand A interacts with strand B and the (A,B) pair has an interleave value of 4, while strand B also interacts with strand C and that (B,C) pair has an interleave value of just 1. With a SMURFLite threshold of 2, the (A,B) pair would be discarded, but the (B,C) pair would remain in the training data. Thus, interleave numbers are properties of *pairs* of β-strands; a β-strand may be involved in multiple pairings, each of which may have a distinct interleave value. Discarding β-strand pairs whose interleave value equals or exceeds the threshold guarantees that the MRF will have no β-strand pairs greater than or equal to that threshold, and thus bounds the computational complexity, which is exponential in the maximum interleave value found in a training template. Figure 3.4 illustrates which β-strand pairs would be removed for two different topologies.

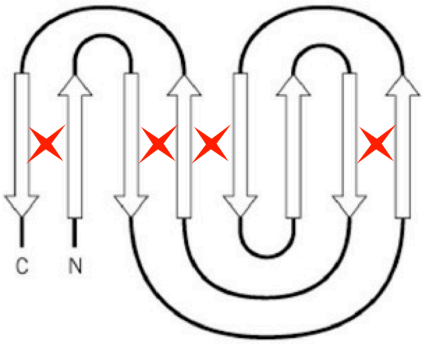
Note that SMURFLite with an interleave threshold of 0, which will discard all β-strand pair information, is simply an HMM.

3.2.7 HMMER implementation

SMURFLite was tested against HMMER version 3.0a2 with the "–seqZ 1" and "–seqE 10000" options applied to hmmsearch, and the "–symfrac 0.2" and "–ere 0.7" options applied to hmmbuild. The –seqZ 1 option ensures that E-values are



(a)



(b)

Figure 3.4: **(a)** A "greek key" β-sheet, indicating which β-strand pairs would be removed by SMURFLite with an interleave threshold of 2. **(b)** A "jelly roll" β-sheet, indicating which β-strand pairs would be removed by SMURFLite with an interleave threshold of 2.

comparable regardless of the size of the sequence database, while the –seqE 10000 option forces HMMER to return results for all query sequences. The –symfrac 0.2 option requires that only 20% of sequences need to be in agreement to cause a match state in a given column (the default is 50%). Given the remote homology at which we were performing experiments, 50% was an unreasonably high threshold that led to few match states being found. This option was also used by [KC09]. The –ere option sets the minimum relative entropy per position target to 0.7 bits (the default is 0.59). Note that HMMER versions 3.0a2 and 3.0 both use SAM sequence entropy ([KBH98]) by default. This entropy weighting scheme has been shown to be superior for remote homology detection tasks ([KC09, Joh06]).

HMMER 3.0a2 was used despite having been superseded by version 3.0, be-

cause it uniformly performs better on this task. This is because version 3.0 contains computational optimizations that cause it to reject a sequence (with no score provided) quickly if it does not appear to align well. These optimizations, however, cause nearly all query sequences outside the family level of homology to fail and return no score, with the result that HMMER version 3.0 never surpasses an AUC of 0.5.

3.2.8 RAPTOR implementation

SMURFLite was tested against RAPTOR, which was run with the options "-a nc" indicating that the default threading algorithm described in the RAPTOR paper ([XLKX03]) was used. In addition, RAPTOR used the weighting parameters "weightMutation = 1.4009760151," "weightSingleton = 1," "weightLoopGap = 16.841836238," "weightPair = 0," "weightGapPenalty = 1," "weightSStruct = 3.0137849223." RAPTOR uses both sequence and structural features, and these options represent the recommended balance of these features ([XLKX03]).

3.2.9 HHPred implementation

SMURFLite was tested against HHPred version 1.5.1. HHPred HMMs for each SCOP family were downloaded from the HHPred web site, and queried using hhsearch. The score of the best-scoring family HMM within each superfamily was used in computing ROC curves.

3.2.10 Whole-genome search

All 1852 protein sequences from *Thermotoga maritima* were queried against β-structural templates constructed from the nr-PDB ([BBB$^+$00]) with non-redundancy determined by an E-value of 10^{-7}, organized according to those 207 β-structural superfamilies from SCOP that were able to be aligned using the Matt structural alignment program, using SMURFLite with an interleave threshold of 2 and simulated evolution mutation rate of 50% on the residues that participate in β-strands. We computed p-values and alignments for all 1852×207 possible hits.

3.3 Results

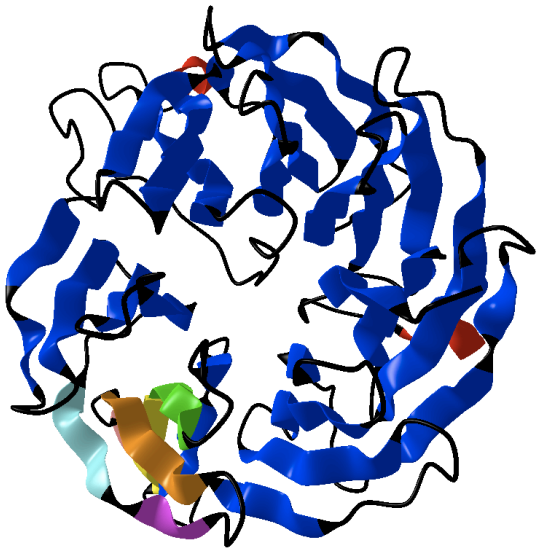
3.3.1 SMURFLite Validation

SMURFLite's ability to recognize β-propellers and barrels was compared to HMMER ([Edd98]), RAPTOR ([XLKX03]), and HHPred ([SBL05]) in a stringent cross-validation experiment, as explained in Section 3.2.2.

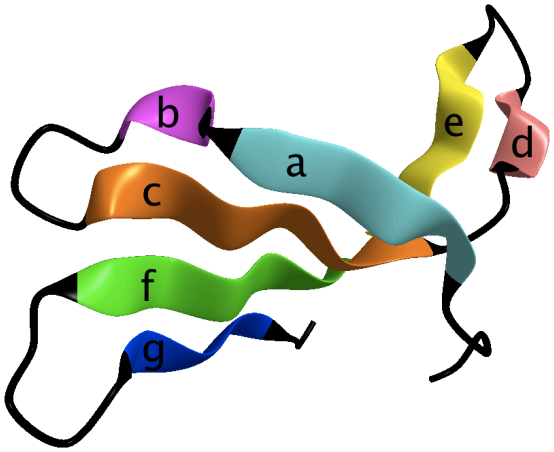
SMURFLite was tested on these 5 propeller folds and 11 barrel superfamilies, with *interleave* thresholds of 1, 2, and 3, and with and without simulated evolution on the β-strands ([KC10]). Here the interleave threshold is a parameter of SMURFLite that trades off the computational complexity with the ability of the MRF to capture complicated long-range dependencies.

The balance between accuracy and computational efficiency is determined by the interleave threshold at which SMURFLite is run. In particular, we found that SMURFLite set to an interleave threshold of 3 or less was always fast. Thus, our first question is how SMURFLite with and without simulated evolution performs on our test set when the interleave threshold is set to 3 or less. We found that SMURFLite became slow at an interleave threshold of 4, and essentially intractable at an interleave threshold of 5 or above. While SMURFLite with an interleave threshold of 1 or 2 requires roughly 1 second of wall-clock time on a 12-core 2.4GHz AMD Opteron server, an interleave threshold of 4 raises this run-time requirement to 7-10 minutes. Restricting the interleave threshold to 3 or less has different impacts on the different folds in our test set. In particular, the β-strands in the propeller folds never have an interleave greater than 3, which means that full SMURF, as we know, is tractable on these folds. However, we were still interested in how simplifying the random field to an interleave of 2 or 1 would impact performance, and also whether simulated evolution would help. In contrast, the barrel superfamilies in our test set contain a maximum β-strand interleave of between 4 and 8. Interestingly, none of these barrels contained any β-strands with an interleave of 3 in the consensus Matt ([MBC08]) alignment, so our restriction of running SMURFLite with an interleave threshold of 3 or less is equivalent, on the barrels, to running SMURFLite with an

interleave threshold of 2. In other words, running the interleave-threshold filter at a threshold of 3 produced identical training data to running it at a threshold of 2.



(a) Full structure of a 7-bladed β-propeller



(b) The most complicated propeller blades have an interleave of 2. Detail of one blade from the structure above, with individual β-strands labeled a through g in sequential order. The interleave values are as follows: (a,c): 2; (b,c): 1; (d,e): 1; (f,c): 3; (f,g): 1.

Figure 3.5: A 7-bladed β-propeller, "Quinohemoprotein amine dehydrogenase" B chain from *Paracoccus denitrificans.*

SMURFLite with interleave threshold 2 and simulated evolution performs well on all propeller folds, with AUCs between 0.89 and 0.99. It always performs better than HMMER, and better than RAPTOR and HHPred except on the 7-

bladed propellers (of which there are 39 non-redundant solved structures in 19 SCOP families), where HHPred achieves an AUC of 0.99 and RAPTOR achieves an AUC of 0.95 versus an AUC of 0.93 for SMURFLite with interleave threshold 2 and no simulated evolution (see Table 3.1). Interestingly, on the 5-bladed propellers (of which there are only 14 non-redundant solved structures in 7 SCOP families), adding simulated evolution seems to greatly improve performance; even SMURFLite with an interleave threshold of 2 with simulated evolution outperforms full-fledged SMURF. While these results focus on the accuracy of the MRF score for the remote homolog decision problem, as opposed to the question of alignment quality, we note that SMURFLite with an interleave threshold of 1 or 2 produces highly similar alignments to full SMURF, particularly with respect to placing the "blades" of the 6-, 7-, and 8-bladed propellers.

For all 11 β-barrel superfamilies, there is a maximum interleave number that ranges from 4 (as in the "Sm-like ribonucleoproteins") to 8 (as in the "Cyclophilin-like" superfamily). We find that for 6 of the 11 β-barrel superfamilies, SMURFLite with an interleave threshold of 2 and simulated evolution outperforms HMMER, RAPTOR, and HHPred. For two of the remaining superfamilies, HMMER performs best, for two of the remaining superfamilies, RAPTOR performs best, and for one superfamily, HHPred performs best (see Table 3.2).

As discussed above, SMURFLite begins to test the limits of computational tractability when interleave numbers of 4 are allowed. Since many barrel structures had β-strand pairs with interleaves of 4, we wished to test if incorporating these more long-range pairwise dependencies into our MRF would improve performance. Some barrel superfamilies on which we tested have only strand pairs of interleave 1 or 2, excepting a pair of β-strands that close the barrel and thus have an interleave equivalent to the number of strands in the barrel. Certainly, including that last strand is beyond the computational power of SMURFLite. Other barrels, whether open or closed, have more complex strand topology and interleaves of 3 or 4 are common even in the middle of the barrels. We chose to run SMURFLite with an interleave of 4 on one of the barrel superfamilies of moderately complex topology,

Table 3.1: AUC on β-Propeller folds

	HMMER	RAPTOR	HHPred	SL1	SL1E	SL2	SL2E	SL3	SL3E
5-bladed	-	-	-	0.75	**0.89**	0.73	**0.89**	0.73	**0.89**
6-bladed	0.82	0.82	0.88	0.92	0.93	**0.96**	0.95	**0.96**	**0.96**
7-bladed	0.89	0.95	**0.99**	0.92	0.91	0.93	0.91	0.93	0.91
8-bladed	-	0.64	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**

Note: for SMURFLite, the number (1,2,3) indicates the interleave threshold, and SEv is simulated evolution. A dash ('-') in a result entry indicates the method failed on these structures, i.e. an AUC of less than 0.6. For issues of space, we abbreviate the SMURFLite entries. For example, SL1 indicates SMURFLite with an interleave threshold of 1, while SL3E indicates SMURFLite with an interleave threshold of 3, augmented by simulated evolution.

the "Barwin-like endoglucanase" superfamily, of which an example appears in Figure 3.2. The "Barwin-like endoglucanase" superfamily contains "Barwin," a protein that may be involved in a common defense mechanism in plants ([SSH$^+$92]).

On the "Barwin-like endoglucanase" superfamily, we find an enormous improvement in performance from capturing that last strand pair, with AUC improving from 0.63 for SMURFLite with an interleave threshold of 2 and simulated evolution, to 0.94 for SMURFLite with an interleave threshold of 4 and simulated evolution (see Figure 3.6). Note that both HMMER and RAPTOR fail entirely on this superfamily, achieving an AUC of less than 0.5.

3.3.2 SMURFLite on Whole Genomes

We considered all 1852 genes from the bacterium *Thermotoga maritima*, a thermophilic organism that bears some similarity to Archaea and whose cell is wrapped in an outer membrane, or "toga" ([HLKT86]). Out of 354 total superfamilies within the SCOP class "All beta proteins", 288 (81%) of which contain at least two protein chains, 207 superfamilies (71%) were structurally consistent enough to be aligned using the Matt ([MBC08]) structural alignment program. We built SMURFLite

Table 3.2: AUC on β-Barrel superfamilies

	HMMER	RAPTOR	HHPred	SMURF-Lite 1	SMURF-Lite 1, SimEv	SMURF-Lite 2	SMURF-Lite 2, SimEv
SMURFLite performs best							
Translation proteins	-	-	0.66	**0.93**	0.92	**0.93**	**0.93**
Barwin-like endoglucanases	-	-	0.75	-	**0.77**	-	0.63
Cyclophilin-like	0.67	0.61	0.7	0.82	**0.85**	0.82	0.83
Sm-like ribonucleoproteins	0.73	0.71	0.77	0.76	0.71	0.76	**0.85**
Prokaryotic SH3-related domain	0.81	-	-	**0.83**	0.82	**0.83**	**0.83**
Tudor/PWWP/MBT	0.78	0.74	0.67	0.83	0.77	**0.83**	0.79
Nucleic acid-binding proteins	0.75	-	0.67	0.76	0.89	0.76	0.92
HHPred performs best							
Translation proteins SH3-like	0.83	0.81	**0.86**	0.62	-	0.62	-
RAPTOR performs best							
PDZ domain-like	0.96	**1.0**	0.99	0.97	0.97	0.97	0.97
FMN-binding split barrel	0.62	**0.82**	0.61	-	-	-	-
HMMER performs best							
Electron Transport accessory proteins	**0.84**	-	0.77	0.63	-	0.63	0.66

Note: for SMURFLite, the number (1,2) indicates the interleave threshold, and SimEv is simulated evolution. A dash ('-') in a result entry indicates the method failed on these structures, i.e. an AUC of less than 0.6

templates for these 207 superfamilies, and obtained from UniProt the protein sequences for each of 1852 genes. We predict 139 of the 1852 genes from *Thermotoga maritima* to belong to one of the 207 β-structural SCOP superfamilies we consider, with a p-value of less than 0.005. Of the 139 genes about which we make predictions, 28 already have solved structures in the PDB, however, since there is a substantial time lag before new PDB structures are assigned to SCOP, only one of those structures was already given a SCOP assignment (and thus only one of these 28 structures potentially informed SMURFLite training). Thus, determining the correct SCOP assignments of the remaining 27 (an easy computational problem given full structural information) allows us to estimate the accuracy of SMURFLite predictions on these structures. Using the Matt ([MBC08]) structural alignment program and the methodology of ([DKCM11]), we computed SCOP superfamilies

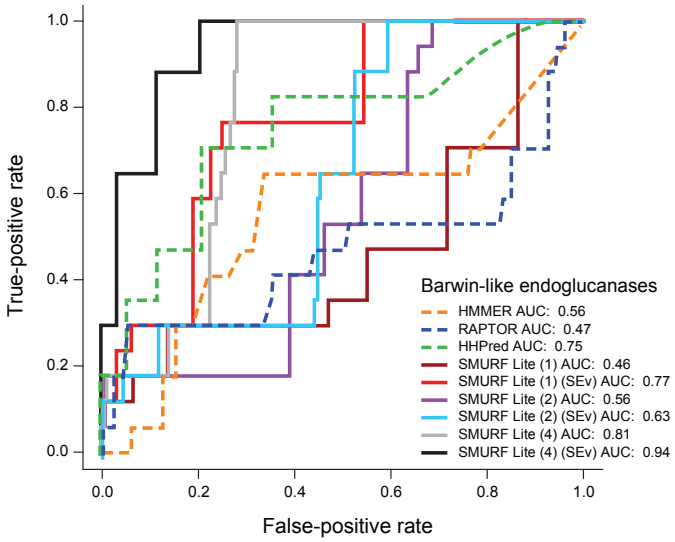


Figure 3.6: Performance of SMURFLite compared to other methods on the "Barwin-like endoglucanases" β-barrel superfamily according to the AUC (Area Under Curve) measure. For SMURFLite, the number (1,2,4) indicates the interleave threshold (indicating which strand pairs in the barrel participate in the MRF; note that interleave 3 is omitted since it is identical to interleave 2 for this fold), and SimEv indicates that simulated evolution was also performed on the β-strands in the training data. As the interleave threshold increases and the MRF becomes more powerful, performance tends to improve. Including simulated evolution also improves performance.

for all 27, and in 100% of the cases, SMURFLite's predictions matched the structural alignments and hence SCOP superfamily assignments. We now survey the remaining 111 structures on which SMURFLite makes predictions, for which structural information is not yet available. 8 of these 111 structures also had their SCOP superfamilies predicted in the study of [ZTW$^+$09] and in all 8 cases, our predictions are in agreement with the other study. We note that for most of these 111 structures, not only is there no solved structure, but there is also no close homology to proteins of solved structure. In particular, none have BLAST hits in UniProt with solved structure and greater than 80% sequence identity, 18 have BLAST hits in UniProt with solved structure and between 30% and 80% sequence identity, and 4 have BLAST hits in UniProt with solved structure and less than 20% sequence identity. As an example, the gene Q9X087 shares only 20% sequence identity with its closest structurally-solved BLAST hit (Rhoptry protein from *Plasmodium yoelii yoelii*, which forms an α-helical structure) but we predict it to belong in the "beta-

Galactosidase/glucuronidase domain" SCOP superfamily with a p-value of 0.0006.

All models predicted can be found at `http://smurf.cs.tufts.edu/smurflite/`

3.4 Discussion

We have presented SMURFLite, a method that combines long-range pairwise β-strand interactions via a simplified Markov random field with simulated evolution, a method that augments training data to capture pairwise β-strand interactions as well. SMURFLite in most cases performs considerably better than HMMER and RAPTOR; however, we examine those structures for which this is not so. We postulate that RAPTOR performs best in the case when there is significant structural conservation across families, whereas HMMER excels when there is a small but highly conserved sequence signature in members of a superfamily. In all four β-barrel superfamilies on which RAPTOR achieves an AUC of less than 0.5, we see considerable structural variation in the protein backbones within each superfamily, according to the metric discussed in Chapter 2, as compared to the other barrel superfamilies. In contrast, the barrels on which RAPTOR performed best exhibited little structural variation.

The cases in which SMURFLite performs poorly exhibit an interesting property: the structural alignment of the protein chains used in the training set preserves few, or sometimes none, of the β-strands as "consensus" β-strands. When a significant number of β-strands are missing in this manner from the training data, SMURFLite exhibits poor specificity, scoring some non-homologous sequences comparably to homologous ones. The "Translation Proteins SH3-Like Domain," a superfamily in which HMMER significantly outperforms SMURFLite, is one in which the consensus alignment obtained from Matt retains zero β-strands, even though each individual structure has four strands. Thus, SMURFLite behaves like HMMER, except without HMMER's heuristic for quickly failing bad alignments, leading SMURFLite to report more false positives.

The very premise of SMURFLite rests on the conservation of β-strands, and

this finding emphasizes the importance of evolutionarily faithful structural alignments. In future work, we will also consider alternative structural aligners, such as TMalign ([ZS05]), in cases where they produce alignments that better conserve secondary structure.

We also compared SMURFLite to HHPred, though in a sense this is not an entirely fair comparison, because HHPred uses *all* of protein sequence space to build profiles for training; thus it can leverage a much larger training set than HMMER, RAPTOR, or SMURF or SMURFLite. Thus it is somewhat surprising that SMURFLite outperforms HHPred in median AUC on the propellers and barrels. We expect HHPred to excel in particular on superfamilies and folds with a high HHPred NEFF ([Söd05]), where NEFF is the "number of effective families" available for training the HHPred HMM. NEFF is a measure of the information-theoretic entropy among a set of sequences; the greater the sequence diversity of such a set, the greater the NEFF.

In contrast, simulated evolution seems to help SMURFLite most on those structural motifs where the HHPred NEFF is lowest; i.e. it can generate diverse training data when diverse training data is lacking. A profile version of SMURFLite would be close in spirit to HHPred, and based on the previous discussions we would expect profiles might improve performance; this will be a subject for future investigation. We observed that simulated evolution either improves or does not affect AUC for β-barrel superfamilies and β-propeller folds with a HHPred NEFF of 20 or lower. The only cases in which we observed simulated evolution decreasing AUC were those cases where the NEFF was greater than 20.

While the intent of using simulated evolution in conjunction with simplified MRFs is to compensate for the removal of highly-interleaved β-strand pairs required for computational feasibility, we find that simulated evolution can still improve full-fledged SMURF in cases of sparse training data. For instance, the 5-bladed β-propellers have only three superfamilies in SCOP, two of which contain only one family. We find that for the 5-bladed β-propeller fold, combining SMURF and simulated evolution improves AUC from 0.73 for full SMURF alone to 0.89.

It is worth noting that simulated evolution on a simple *pointwise* basis, as implemented by Kumar and Cowen [KC09], could likely be incorporated into the hidden Markov itself as a set of Dirichlet mixture priors. However, it is not clear how the *pairwise* model could be incorporated. In addition, we determine β-strand paired residues on the *full* Markov random field, before removing any pairing information. Thus, in this case, simulated evolution may be mitigating the loss of this β-strand pairing information.

We have demonstrated that SMURFLite is a powerful MRF methodology for β-structural motif recognition that is computationally tractable enough to scale to whole genomes, requiring approximately three hours to scan the *Thermotoga maritima* genome on a small compute cluster. We have also shown that increasing the interleave number for SMURFLite can have dramatic effects on performance, but at a great computational cost. Methods that allow us to retain all β-strand pairs, such as loopy belief propagation[Pea88] or stochastic search, merit investigation. As our dependency graph is not a tree, loopy belief propagation may present difficulties with convergence and inexact inference. Nonetheless, looking at heuristic methods ([SHJ97, WJ99]) that approximately compute the SMURF score more efficiently may add even more power to our approach in practice.

Chapter 4

Protein Remote Homology Detection Using Markov Random Fields and Stochastic Search

4.1 Introduction

In Chapter 3, we explored a method for simplifying the computational complexity of the Markov random field, as well as an approach, called "simulated evolution," for mitigating the loss in accuracy resulting from this simplification. We showed that this approach, called SMURFLite, outperformed several existing methods at remote homology detection in β-barrels and propellers.

In this work, we demonstrate another approach for computing the SMURF energy function for remote homology detection. Building upon the β-structural Markov random field templates from SMURF and SMURFLite, we demonstrate a method for remote homology detection that does not discard any β-structural information, and yet remains computationally tractable on any protein structure.

We have developed MRFy, an algorithm that relies on stochastic search to

find a near-optimal parse of a query sequence onto the SMURF Markov random field. We also provide an implementation of MRFy, written in the Haskell functional programming language; this implementation is discussed in our "experience report" on computational biology software in Haskell [DGR12], which is not part of this dissertation.

We test MRFy on the same set of barrel folds in the mainly-β class of the SCOP hierarchy as was used to test SMURFLite in Chapter 3, in stringent cross-validation experiments. We show a mean 0.4% (median 1.7%) improvement in Area Under Curve (AUC) for β-structural motif recognition as compared to the SMURFLite results in Chapter 3 and [DHBC12]. By these same benchmarks, we show a mean 5.5% (median 16%) improvement over HMMER ([Edd98]) (a popular HMM method), a mean 29% (median 16%) improvement as compared to RAPTOR ([XLKX03]) (a well-known threading method), and a mean 13% (median 14%) improvement in AUC over HHPred ([Söd05]) (a profile-profile HMM method).

4.2 Methods

4.2.1 Markov random field model

MRFy builds on the SMURF and SMURFLite Markov random field model, as discussed in Chapter 3, which uses multidimensional dynamic programming to simultaneously capture both standard HMM models and the pairwise interactions between amino acid residues bonded together in β-sheets.

In particular, the "Plan7" hidden Markov model is modified to represent hydrogen-bonded β-strands with additional, non-local edges. Because the β-strands in a SMURF or MRFy template represent *consensus* β-strands, those present in at least some fraction (in our experiments, at least half) of the sequences participating in the training alignment, we prohibit insertions and deletions in those strands. Thus, we collapse those nodes of the "Plan7" model to be just match states; the transitions to insertion and deletion states are removed. Figure 4.1 illustrates this architecture.

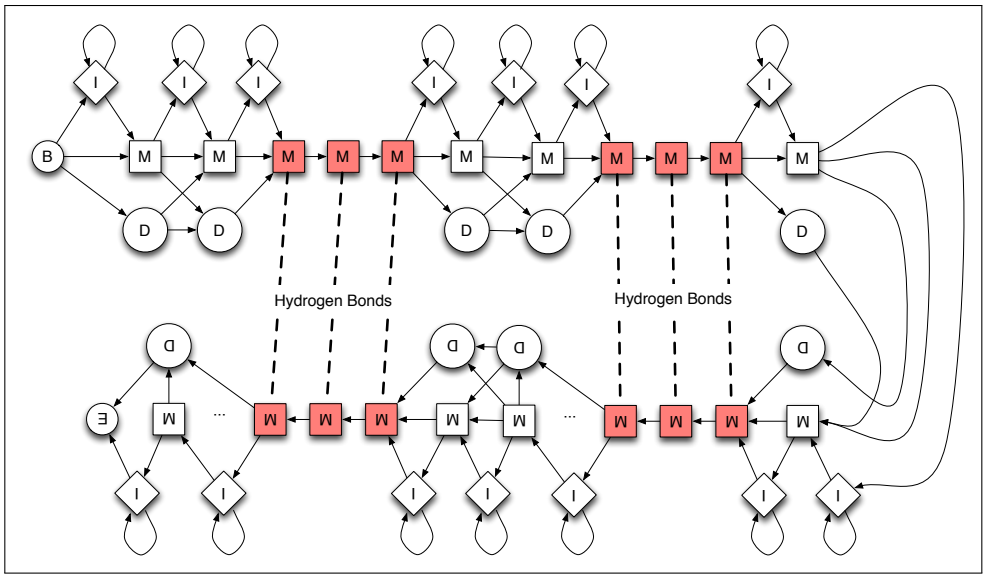


Figure 4.1: A Markov random field with two β-strand pairs

Recall from 1 that the standard form of the Viterbi recurrence relations for computing the most likely path of a sequence through a hidden Markov model is:

$$V_j^M(i) = \frac{e_{M_j}(x_i)}{q_{x_i}} \times \max \begin{cases} V_{j-1}^M(i-1) \times a_{M_{j-1}M_j} \\ V_{j-1}^I(i-1) \times a_{I_{j-1}M_j} \\ V_{j-1}^D(i-1) \times a_{D_{j-1}M_j} \end{cases}$$

$$V_j^I(i) = \frac{e_{I_j}(x_i)}{q_{x_i}} \times \max \begin{cases} V_j^M(i-1) \times a_{M_j I_j} \\ V_j^I(i-1) \times a_{I_j I_j} \end{cases} \tag{4.1}$$

$$V_j^D(i = \max \begin{cases} V_{j-1}^M(i) \times a_{M_{j-1}D_j} \\ V_{j-1}^D(i) \times a_{D_{j-1}D_j} \end{cases}$$

In the SMURF or MRFy Markov random field model, we add non-local interactions to these probabilities, resulting in conditional probabilities. When column j of an alignment is part of a β-strand and is paired with another column $\pi(j)$, the probability of finding amino acid x_i in column j depends on whatever amino acid x' is in column $\pi(i)$. If x' is in position i' in the query sequence, Viterbi's equations are

altered; for example, $V_j'^M(i)$ depends not only on $V_{j-1}'^M(i-1)$ but also on $V_{\pi(j)}'^M(i')$. The distance between j and $\pi(j)$ can be as small as a few columns or as large as a few hundreds of columns. Because $V_j'^M(i)$ depends not only on nearby values but also on $V_{\pi(j)}'^M(i')$, we must modify the Viterbi recurrence relations.

Note that hydrogen-bonded β-strand residues may only occupy match states in the Markov random field, so only the corresponding terms of the recurrence relation need be modified. The revised Viterbi recurrence relation for the Markov random field is:

$$V_j^M(i \;=\; \frac{e_{M_j}(x_i)}{q_{x_i}} \times \max \begin{cases} V_{j-1}^M(i-1) \times a_{M_{j-1}M_j} \times P(x_i|x_{\pi j}) \\ V_{j-1}^I(i-1) \times a_{I_{j-1}M_j} \times P(x_i|x_{\pi j}) \\ V_{j-1}^D(i-1) \times a_{D_{j-1}M_j} \times P(x_i|x_{\pi j}) \end{cases} \tag{4.2}$$

where $x_{\pi j}$ represents the amino acid in column πj, which is hydrogen-bonded to the amino acid x_i in column j.

For reasons of convenience, as well as avoiding floating-point underflow due to exceedingly small numbers, we typically work in negative log space. Since a probability can range from 0 to 1, the log of a probability must be a negative number, and thus the negative log of that probability is a (small) positive number. Each probability is transformed into its negative log, resulting in the final form:

$$V'^{M}_{j}(i = e'_{M_j}(x_i) + \min \begin{cases} a'_{M_{j-1}M} + V'^{M}_{j-1}(i-1) + P'(x_i|x_{\pi j}) \\ a'_{I_{j-1}M_j} + V'^{I}_{j-1}(i-1) + P'(x_i|x_{\pi j}) \\ a'_{D_{j-1}M} + V'^{D}_{j-1}(i-1) + P'(x_i|x_{\pi j}) \end{cases}$$

$$V'^{I}_{j}(i) = e'_{I_j}(x_i) + \min \begin{cases} a'_{M_j I} + V'^{M}_{j}(i-1) \\ a'_{I_j I_j} + V'^{I}_{j}(i-1) \end{cases} \tag{4.3}$$

$$V'^{D}_{j}(i = \min \begin{cases} a'_{M_{j-1}D} + V'^{M}_{j-1}(i) \\ a'_{D_{j-1}D} + V'^{D}_{j-1}(i) \end{cases}$$

given the transformations:

$$\begin{aligned} a'_{s\hat{s}} &= -\log a_{s\hat{s}} \\ e'_s(x) &= -\log \tfrac{e_s(x)}{q_x} \\ V'^{M}_{j}(i) &= -\log V^{M}_{j}(i) \\ P'(x_i|x_{\pi j}) &= -\log P(x_i|x_{\pi j}) \end{aligned} \tag{4.4}$$

This is exactly the recurrence relation that SMURF [MBC10] and SMURFLite [DHBC12] solve using multidimensional dynamic programming. As demonstrated in Chapter 3, as the *interleave* of the β-strands increases, the computational complexity grows exponentially.

As an alternative to solving these more complex recurrence relations, we might consider a divide-and-conquer approach. Each β-strand can be thought of as breaking the larger model into two smaller models; collectively, all the β-strands divide the Markov random field into many small, *independent* hidden Markov models. Thus, for any particular path through the Markov random field, corresponding to a particular placement of query sequence residues onto the nodes of the model, we could compute the augmented Viterbi score by summing the Viterbi scores of each smaller hidden Markov model, along with the contribution to the Viterbi score from

the β-strands.

Since only match states are allowed for β-strand residues, the contribution of each such residue is only:

$$V'^{M}_{j}(i) = e'_{M_j}(x_i) + a'_{M_{j-1}M} \; + V'^{M}_{j-1}(i-1) + P(x_i|x_{\pi j}) \tag{4.5}$$

The asymptotic complexity of the Viterbi algorithm is $O(mn)$, where m is the length of the model and n is the length of the query sequence. Furthermore, the asymptotic complexity of the beta-strand contribution to the Viterbi score for a particular placement of residues is just $O(b)$, where b is the combined length of the β-strands.

Thus, a new algorithm for computing the optimal path through a Markov random field for a given query sequence presents itself. Since we require that every β-strand position be occupied by a residue (as we force those positions into match states), we could simply consider every possible assignment of a residue to a β-strand, computing the score for each one, and choose the best-scoring placement.

Metaphorically, we can picture the residues of the query sequence as beads, and the Markov random field as the string of a necklace. The β-strands can be thought of as particular substrings of the string that must be covered by beads, while non-β regions may be exposed (resulting in delete states in the model). To continue the metaphor, we may force extra beads onto non-β regions of the string, resulting in insert states in the model. Given that the beads already have a specified order, we must consider all the ways to slide the beads up and down the string such that all of the β-regions are covered. Since the regions between β-strands can have their contribution to the score computed according to the Viterbi recurrence relations, we need only consider all the unique ways to assign residues to the β-strand nodes.

4.2.2 Proof that the model is exponential in complexity

Here, we prove that there are an exponential number of possible β-strand placements that must be considered.

Definition Let a Markov random field model (N, B) be defined as a sequence N of nodes $n_i, i \in (1..m)$, and a sequence B of β-strands $b_i, i \in (1..k)$. Each β-strand has length l_i, and contains a subsequence of the nodes N. This subsequence is determined by the specifics of the model, which can be referred to as $b_{ij}, i \in (1..m), j \in (1..l_i)$. Let a query sequence be defined as a sequence R of residues $r_i, i \in (1..n)$.

Definition Let $L = \sum_{i,i<=k} l_i$.

Lemma 4.2.1 *Given a model (N, B) and a query sequence R, L residues are placed in β-strands.*

Proof Because each β-strand b_i must be populated by exactly l_i residues, $\forall j, j > 1$, b_{ij} is uniquely determined by the sequence R. For each β-strand position b_{ij}, one residue is placed. Thus, $\sum_{i,i<=k} l_i$ residues are placed in β-strands.

Theorem 4.2.2 *For a Markov random field (N, B) with k β-strands b_i, each of length l_i, and thus containing positions for residues b_{ij} and a query sequence ri of length n, there are $O(n^k)$ ways to assign residues to the β-strands.*

Proof From the n residues in the query sequence R, we need to place L residues across all B β-strands. We represent this as choosing an index $i \in (1..n)$ for the first position b_{i1} of each β-strand. Since each β-strand b_i consumes l_i residues, this choice for the first β-strand, b_{11}, leaves $n - L - l_i$ possible placements for b_{21}. In practice, β-strands range from two to twelve residues, so to simplify counting, we assume each l_i is simply a maximum length l_{max}. This only decreases the number of possible assignments, yielding a lower bound on the number of placements. Then choosing an index to place on b_{i1}, in general, leaves $n - L - (i \times l_{max})$ choices for $b_{(i+1)1}$. Thus, there are:

$$\prod_{i \in (1..k)} n - L - (i \times l_{max}) = (n - 2L - k \times (l_{max}))^{\lceil \frac{k}{2} \rceil} \tag{4.6}$$

possible placements of R onto (N, B). Asymptotically, as n grows, this is dominated by n^k, leading to an asymptotic complexity of $O(n^k)$. ∎

A typical Markov random field might have 10 or 20 β-strands, and a typical protein query sequence might have between 300 and 600 residues. Thus, if we wish to consider all possible paths through a Markov random field for a protein sequence, we must consider as many as $600^{10} \approx 6 \times 10^{27}$ possible paths through the model. Clearly, this computation can be broken into many parallel parts, but this still poses an intractable problem in many cases.

4.2.3 Stochastic search

Since an exhaustive search for an optimal alignment of a protein sequence to a Markov random field is exponential in complexity, we turn to stochastic search to mitigate this complexity.

Stochastic search encompasses a family of approaches for finding optimal or near-optimal solutions to optimization problems. Stochastic search approaches are promising when a search space is large, so that exhaustive search is prohibitive, and when an optimization problem does not lend itself to analytic solutions. The generic form of stochastic search is that a solution is guessed at and evaluated, and then subsequent guesses are made as refinements to this initial guess, until some termination condition is met. The function used for evaluation is called the *objective function*.

Framed as an optimization problem, MRFy, like SMURF, seeks to minimize the augmented Viterbi score (see Equation 4.3), which equates to maximizing probability (recall that this score is the negative log of a probability). SMURF finds this minimum exactly, using multi-dimensional dynamic programming, which is exponential in the interleave number of beta strands (see Chapter 3). MRFy, in contrast, uses stochastic search, as described next.

Given a placement of query-sequence residues into β-strand nodes of the Markov random field, the score can be computed exactly. Thus, the search space

is the set of all possible ways to place residues on these nodes, as discussed in Section 4.2.2. Many stochastic search techniques rely on a gradient ascent (or descent) approach, which makes moves (or refines guesses) along the steepest gradient, leading quickly to local optima; various heuristics such as simulated annealing [KV83] can then help avoid getting stuck in poor local optima.

However, we know of no way to compute a gradient on the search space of β-strand placements, and so we must take approaches that do not rely on this gradient. Instead, we must rely on a random-mutation model of search, which generates one or more candidate solutions (guesses) from a previous solution, and then evaluates the cost function (in our case, the augmented Viterbi score) to determine whether those guesses are better or worse than the previous step. This can be likened to climbing a hill in the dark, feeling one's way with one foot before committing to a step. This approach is referred to as *random-mutation hill climbing*[Dav91].

In our representation, a particular solution is represented by an ordered list of integers, one integer per β-strand in the Markov random field. The value of each integer indicates the index, in the query sequence, of the residue assigned to the first position of that β-strand. Since the alignments to the regions of the Markov random field are solved exactly by the Viterbi algorithm, this ordered list of integers uniquely represents a solution to a Markov random field.

While the picture we have presented for our Markov random field model is most precisely explained by assigning residue indices to the positions of β-strands, it may be more intuitive to consider the equivalent problem of "sliding" these β-strands along the query sequence. We will use this analogy in the following description of initial guesses.

We explored three models for generating initial guesses for our search techniques:

- *Random-placement model.* First, we implemented a model that uniformly positions the β-strands along the query sequence, under the constraint that only legal placements may be generated, and thus the placement of any β-

strand must leave room for all the other β-strands in the model.

- *Placement based on secondary structure prediction.* Next, we implemented a model that uses the PSIPRED [MBJ00] secondary-structure prediction program to determine the positions of β-strands. Given a PSIPRED prediction for the secondary structure of a query sequence, we place β-strands at the most likely locations according to this prediction profile, randomized by a small amount of noise. The difficulty with this approach is that, while PSIPRED is reasonably accurate when it is allowed to perform PSI-BLAST[AMS$^+$97] queries to build a sequence profile, this comes at a run-time cost that completely dominates the running time of MRFy. However, while non-profile-based PSIPRED predictions are computationally cheap, they provide poor accuracy.

- *Placement based on scaling the template.* Finally, we implemented a model based on the observation that true homologs to a structurally-derived template should have their β-strands in very roughly similar places, in sequence, to the proteins that made up that template. This will not always hold, but appears to provide for reasonable initial guesses. Given the position of each β-strand within a template Markov random field, we scale the query sequence linearly (as it may be shorter or longer than the model) and place the β-strands in scaled positions. Note that we do not scale the β-strands themselves; their lengths are preserved. We scale only the distances between β-strands. We inject a small amount of noise into the placements, so that population-based models, such as multi-start simulated annealing and genetic algorithms, start with heterogenous solutions.

Since we do not know how to determine when a stochastic search process has found a *global* optimum (as opposed to a good local optimum), we must also have some termination criterion for the search. We implemented three alternative termination criteria:

- A simple generation-counting approach, where the search terminates after a user-specified number of generations
- A time-based approach, where the search terminates after a user-specified amount of time has elapsed
- A *convergence* model, where the search terminates after the search has failed to improve after a user-specified number of generations

In practice, these criteria are easily combined, with a convergence approach often halting searches early with good results, while the generation- or time-based limit ensuring that the search does not take longer than a user is willing to wait. We next describe the alternative heuristics that MRFy implements for stochastic search: simulated annealing, a genetic algorithm, and a local search strategy.

4.2.3.1 Simulated Annealing

Simulated annealing [KV83] is a heuristic for stochastic search, inspired by the physical process of annealing in metals. Whereas a simple hill-climbing approach will always move downhill (if the task is minimization) or uphill (if the task is maximization), if the search begins near to a poor local optimum, the search will terminate at that local optimum. Simulated annealing introduces an *acceptance probability function*:

$$P(e, e', T) = \begin{cases} 1, & \text{if } e' < e \\ \exp(-(e' - e)/T), & \text{otherwise} \end{cases} \quad \text{where } e = E(s), \; e' = E(s') \tag{4.7}$$

which relies on some energy function $E(s)$ of the current state s and a candidate state s', and a temperature function T that tends towards zero as the search progresses.

In our implementation, we used an exponentially-decaying temperature function:

$$T(t) = k^t \times T_0 \tag{4.8}$$

given time t, initial temperature T_0, and a constant k. The motivation for this decaying temperature function is that, as time progresses, the likelihood of being in a *poor* local optimum lessens, and thus, the closer to random hill-climbing we would like the search to behave.

Our energy function $E(s)$ is, naturally, the augmented Viterbi score of a placement:

$$E(s) = V'^{M}_{m}(n) \tag{4.9}$$

where m is the final residue in the query sequence and n is the final node in the Markov random field, and the β-strand placements are determined by s.

We implemented simulated annealing in MRFy according to this model. We also implemented a *multi-start* version of simulated annealing in MRFy, where a set of independently-generated guesses is subject to simulated-annealing random descent, in parallel. At the termination of the search, the best solution from among all the candidates is chosen.

4.2.3.2 Genetic Algorithm

A genetic algorithm [HR77] is a search heuristic inspired by biological evolution. A genetic algorithm relies on the idea of *selection* among a population of varied solutions to an optimization problem. At each of many generations, the fitter individuals in the population–those solutions which exhibit more optimal scores–are allowed to continue into the next generation. Not only do they continue into the next generation, but they are allowed to "reproduce," or recombine, to produce new solutions. A particular solution to a problem, within the context of a genetic algorithm, is called a *chromosome*. At each generation, some fraction of the fittest solutions are selected and randomly paired with one another. Each pair of solutions

produces one or more offspring; each offspring is the result of two steps: *crossover* of the two chromosomes, followed by random *mutation* of the offspring. The mutation is nondeterministic; the crossover may be deterministic or nondeterministic. The resulting offspring, along with their parents, are then evaluated according to the objective function, and this process iterates until some termination condition.

MRFy's genetic algorithm implementation uses the same representation for a placement as simulated annealing: an ordered list of integers.

Let a *placement* p on a model with k β-strands be an ordered set of integers $p_i, i \in (1..k)$. Given two placements, p and q, MRFy implements crossover of two chromosomes using the following algorithm:

1. Set the new placement, p', to the empty set.
2. Repeat until all placements have been chosen:
   (a) Let $p'_0 = p_0$
   (b) Let $p'_k = q_k$
   (c) Remove p_0 and p_k from p
   (d) Remove q_0 and q_k from q
   (e) $p =< p_1, ..., p_{k-1} >$
   (f) $q =< q_1, ..., q_{k-1} >$

Our actual implementation is purely functional, and simply consumes elements from lists. In effect, though, this algorithm simply chooses the 'left-most' elements from one parent and the 'right-most' elements from another. After crossover, the mutation step simply moves each element p_i of the placement p by a small, random amount, within the constraints imposed by the neighboring β-strands. The motivation behind this approach is to take two solutions that are of high fitness (recall that the worst solutions at every generation are not allowed to contribute to the next generation), and produce a new solution that combines one "half" (roughly) of one solution with one "half" of the other. See Figure 4.2 for an illustration of this procedure.

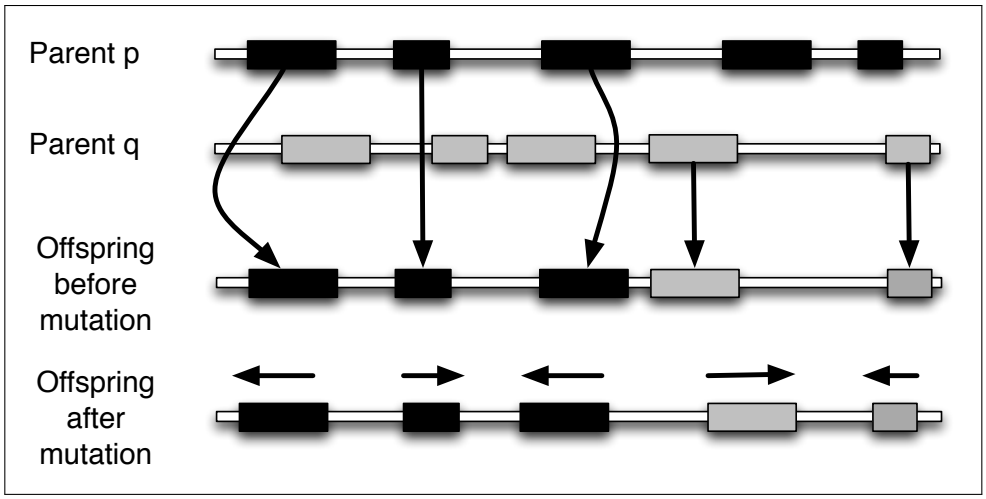


Figure 4.2: The crossover and mutation process in MRFy's genetic algorithm implementation. Given parent p (black) and parent q (gray), alternate left and right placements from p and q. Then, apply small random mutations to the resulting placement p'.

Given these operations for crossover and mutation, MRFy's genetic algorithm implementation initializes a population of a user-specified size P (typically one thousand placements, though we experimented with as many as ten thousand). In parallel, each placement is scored according to the objective function. Since scoring is far more computationally expensive than crossover and mutation, we allow them all to reproduce, paired at random. We then score them, and choose the P best-scoring placements for the next generation. This process repeats until a termination condition is met, at which point the single best placement is returned. We note that a future enhancement to MRFy could return the k best placements for some user-specified threshold k, if multiple high-scoring alignments were to be considered.

4.2.3.3 Local Search

Constraint-based local search [HM05] is a family of approaches for exploring "neighborhoods" in feature space in a randomized manner, subject to the constraints of that solution space. In the context of MRFy, the constraints are the previously-discussed restrictions that β-strands cannot overlap, and every residue must be

placed in a β-strand. The motivation for local search is, in a particularly uneven fitness landscape, hill climbing will often reach nearby local optima. Thus, given a single candidate solution, local search explores the immediate neighborhood in great detail (perhaps, but not necessarily exhaustively). When the local search cannot escape a local optimum, then some sort of *non-local* move may be attempted.

This non-local move may rely on a population-based diversification approach, in which parts of the solution may change dramatically. In a sense, local search bears some resemblance to a genetic algorithm, except that a population of solutions is created only when the search is stuck in a local optima, and the best solution in that population is chosen for a new search.

In MRFy's implementation, each step in the search consists of two phases: *diversification* (See Figure 4.3) and *intensification*. The diversification algorithm is as follows:

- Begin with a candidate solution s (a placement), which is just an ordered list of integers.
- Given s, break the list into three sub-lists s_0, s_1, s_2, at randomly-chosen boundaries.
- Choose one of the sub-lists s_i at random, and mutate it into k copies s_{i1} through s_{ik} at random, for some user-defined value of k (we used $k = 10$), within the constraints imposed by the other sub-lists and the lengths of the β-strands.
- Re-combine each set of lists, (s_{1j}, s_{2j}, s_{3j}) into a new placement $s'_j, j \in (1..k)$.
- Score each placement s'_j, return the best-scoring of the k new placements as a new solution.

Once diversification produces a new candidate solution, intensification brings it toward a local minimum. The intensification algorithm is as follows:

- Begin with a candidate solution s.

- Repeat until no better-scoring placements are generated.
  - For each element $e \in s$, generate four new placements s'_{i1} through s'_{i4} by moving e up and down by 1 and two, as long as those moves do not violate the constraints.
  - Score each candidate placement s'_{ij}.
  - Set s to the best-scoring candidate placement s'_{ij}
- Return s as a new solution.

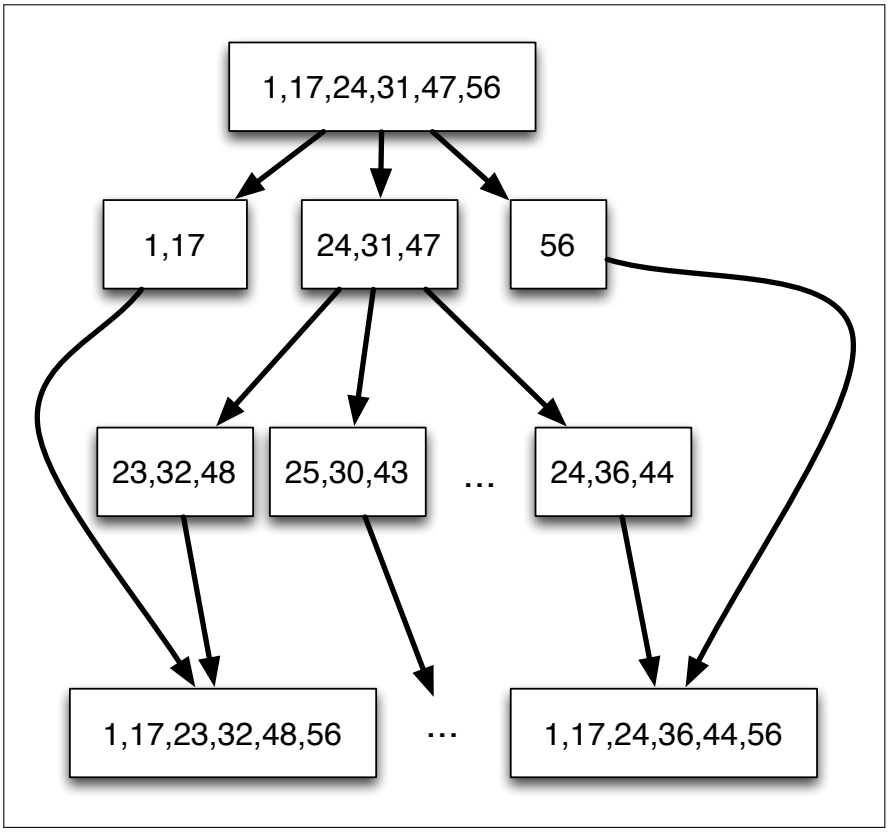


Figure 4.3: The diversification step in local search.

4.2.4 Evaluating search strategies

As MRFy supports three significantly different stochastic search strategies, and a number of tunable parameters such as termination conditions and (for simulated annealing) the cooling schedule, we conducted a search over parameter space using

a small data set. We built Markov random field templates from the fold "8-bladed Beta-Propellers", and the superfamilies "Barwin-like endoglucanases" (a β-barrel superfamily) and "Concanavalin A-like lectins/glucanases" (a β-sandwich superfamily). We were interested in the speed of convergence for a true-positive test case, so we tested each template with a protein sequence chosen from that fold or superfamily: for the 8-bladed propeller, we chose ASTRAL chain d1lrwa_ (Methanol dehydrogenase, heavy chain from *Paracoccus denitrificans*). For the barwin-like endoglucanases, we chose ASTRAL chain d2pica1 (Membrane-bound lytic murein transglycosylase A, MLTA from *E. coli*). For the lectins/glucanases, we chose ASTRAL chain d2sbaa_ (Legume lectin from soy bean (*Glycine max*)).

We tested simulated annealing with a population size of 10, a maximum number of generations of 10000, convergence periods of 200, 500, and 1000 generations, and a cooling factor of 0.99 (preliminary tests showed little impact from varying the cooling factor among 0.9, 0.99, and 0.999).

We tested the genetic algorithm implementation with a population size of 1000 and 10000, a maximum number of generations of 500, and convergence periods of 10, 50, and 100.

Since the local search distinguishes between diversification and intensification, counting the number of generations is ambiguous; we used a time limit of 10 seconds, 30 seconds and 5 minutes. All tests were conducted on a 12-core AMD Opteron 2427 with 32GB RAM, devoting all 12 cores to MRFy. For each test, we report statistics based on ten runs for each set of parameters.

4.2.5 Simulated Evolution

In MRFy, we incorporated precisely the same "simulated evolution" implementation, as first proposed by Kumar and Cowen [KC09, KC10], as we did for SMURFLite in Chapter 3. We added pairwise mutations based on β-strand pairings. Unlike in Chapter 3, here we were not attempting to mitigate the loss of information due to simplifying the Markov random field, but rather attempting to compensate for sparse training data. This was motivated in part by the observation that SMURFLite

benefited most from simulated evolution when the "number of effective families" was low. We use the same mutation frequencies as in Chapter 3. For each artificial sequence, we mutate at a 50% mutation rate per length of the β-strands.

4.2.6 Datasets

From SCOP ([MBH95]) version 1.75, we chose the same β-structural superfamiles as for SMURFLite (Chapter 3). These superfamilies were: "Nucleic acid-binding proteins" (50249), "Translation proteins" (50447), "Barwin-like endoglucanases" (50685), "Cyclophilin-like" (50891), "Sm-like ribonucleoproteins" (50182), "PDZ domain-like" (50156), "Prokaryotic SH3-related domain" (82057), "Tudor/PWWP/MBT" (63748), "Electron Transport accessory proteins" (50090), "Translation proteins SH3-like domain" (50104), and "FMN-binding split barrel" (50475).

4.2.7 Training and testing process

For the β-barrel superfamilies, we performed strict leave-family-out cross-validation. We built training templates at the superfamily level. For each superfamily, its constituent families were identified. Each family was left out, and a training set was established from the protein chains in the remaining families, with duplicate sequences removed. We built an MRF on the training set, both with and without training-data augmentation using the same "simulated evolution" implementation as in Chapter 3: we generate 150 new artificial training sequences from each original training sequence. For each artificial sequence, we mutate at a 50% mutation rate per length of the β-strands. We chose protein chains from the left-out family as positive test examples. Negative test examples were protein chains from all other superfamilies in SCOP classes 1, 2, 3 and 4 (including other barrel superfamilies), indicated as representatives from the nr-PDB ([BBB$^+$00]) database with non-redundancy set to a BLAST E-value of 10^{-7}.

We used MRFy's local search mode (see Section 4.2.3.3) to align each test example to the trained MRF. The score reported for MRFy was the combined HMM and pairwise score from the MRF, which is identical to the SMURF energy

function. For each training set, the scores for both methods (MRFy with and without simulated evolution) were collected and a ROC curve (a plot of true positive rate versus false positive rate) computed. We report the area under the curve (AUC statistic) from this ROC curve ([SKP08]).

4.3 Results

4.3.1 Search strategies

For the three stochastic search approaches, we compared the raw score achieved by each approach under a variety of conditions, as discussed in Section 4.2.4. The raw score is simply the negative log of the probability of the best path found through the model. Thus, raw scores are not comparable between models, but they are comparable between query sequences for a given model.

Table 4.1 indicates the performance of different stochastic search techniques on the 8-bladed β-propeller fold. While the simulated annealing and genetic algorithm approaches exhibit less variance (a smaller standard deviation) from run to run, they do not approach the minimum score of the local search approaches. Multi-start simulated annealing with a population of 10 and a convergence threshold of 200 generations averages 29.3 seconds per search, but only achieves a minimum score of 2112, though it converged in all cases.

In contrast, local search, given 30 seconds, achieves a minimum score of 1982, and even in only 10 seconds achieves a minimum score of 1992. However, the global minimum score of 1781, which is achieved by SMURF on the 8-bladed β-propeller template, is only reached by MRFy with local search two out of ten times, and this result required local search be allowed to run for twenty minutes. Thus, for this problem domain, local search seems to outperform our simulated annealing and genetic algorithm implementations.

Table 4.2 indicates the performance of the stochastic search techniques on the "Barwin-like endoglucanases" β-barrel superfamily. These structures are less complex than the propellers, even though they are *more* computationally complex

for SMURFLite (Chapter 3) if an interleave threshold greater than 2 is used. We see less variance than with the propellers, but once again, the local search technique achieves a lower minimum score than simulated annealing or the genetic algorithm.

Notably, local search achieves a minimum score of 978, which an *exhaustive* search indicates to be a global minimum for this sequence on this template. With a time limit of 10 seconds, local search found this global minimum in one out of ten runs. With a time limit of 30 seconds, local search found it in two out of ten runs, and with a time limit of 5 minutes, in four out of ten runs.

Table 4.1: Stochastic search performance on 8-bladed β-propeller

	Min Score	Mean Score	Std Score	Mean Time (s)
SA 200	2112	2139	12.2	29.3
SA 500	2129	2146	9.3	1020
SA 1000	2112	2130	7.8	3314
GA 1000/10	2105	2126	6.6	285
GA 1000/50	2094	2118	7.7	1239
GA 1000/100	2107	2120	**3.8**	548
GA 10000/10	2087	2111	7.2	5809
GA 10000/50	2094	2112	7.1	5174
GA 10000/100	2079	2114	9.0	10226
LS 10s	1992	2015	19.4	**10**
LS 30s	1982	1991	10.9	30
LS 5m	**1818**	**1876**	37.2	300

Performance of stochastic search techniques on an 8-bladed β-propeller template. SA is Simulated Annealing, GA is Genetic Algorithm, and LS is Local Search. For Simulated Annealing, we show results for convergence thresholds of 200, 500, and 1000 generations. For the Genetic Algorithm, we show results for convergence thresholds of 10, 50, and 100 generations, and for population sizes of 1000 and 10000. For Local Search, we show results for time limits of 10 seconds, 30 seconds and five minutes, on a 12-core AMD Opteron. MRFy never achieved the global optimum score of 1781, achieved by SMURF, on this template, except when local search was given 20 minutes of compute time, in which case it found the global optimum two out of ten times.

Table 4.3 indicates the performance of the stochastic search techniques on the "Concanavalin A-like lectins/glucanases" β-sandwich superfamily. These structures

Table 4.2: Stochastic search performance on "Barwin-like" β-barrel

	Min Score	Mean Score	Std Score	Mean Time (s)	Optimal
SA 200	1064	1071	3.8	79.5	0
SA 500	1047	1063	7.6	104	0
SA 1000	1024	1047	14.0	523	0
GA 1000/10	1061	1069	3.6	232	0
GA 1000/50	1059	1066	3.1	442	0
GA 1000/100	1058	1069	4.0	1382	0
GA 10000/10	1058	1063	2.5	8205	0
GA 10000/50	1059	1061	**2.2**	10306	0
GA 10000/100	1057	1061	**2.2**	16395	0
LS 10s	**978**	995	16.2	**10**	0.1
LS 30s	**978**	987	6.9	30	0.2
LS 5m	**978**	**981**	2.9	300	0.4

Performance of stochastic search techniques on the "Barwin-like endoglucanases" β-barrel template. SA is Simulated Annealing, GA is Genetic Algorithm, and LS is Local Search. For Simulated Annealing, we show results for convergence thresholds of 200, 500, and 1000 generations. For the Genetic Algorithm, we show results for convergence thresholds of 10, 50, and 100 generations, and for population sizes of 1000 and 10000. For Local Search, we show results for time limits of 10 seconds, 30 seconds and five minutes, on a 12-core AMD Opteron. The "Optimal" column indicates the fraction of runs for each search method that achieved the global optimum.

are also more complex than the propellers, even though they are also more computationally complex for SMURFLite with an interleave threshold greater than 2. On this superfamily, there is a closer overlap between the minimum score achieved by simulated annealing, at 790, and the range seen by local search; local search with a time limit of 30 seconds achieves a *mean* minimum score of 791, though its best was 740.

Notably, when given a time limit of 5 minutes, local search achieved the *global minimum* of 554 (as determined by exhaustive search) ten out of ten times. Local search never found this score when given only 10 seconds or 30 seconds as a time limit.

Our Haskell implementation made it exceedingly easy to parallelize MRFy across multiple processing cores. By default, MRFy will take advantage of all processing cores on a system; we tested the parallel speedup on a system with 48 pro-

Table 4.3: Stochastic search performance on β-sandwich

	Min Score	Mean Score	Std Score	Mean Time (s)	Optimal
SA 200	795	834	18.6	84.7	0
SA 500	790	820	17.3	192	0
SA 1000	791	811	14.7	493	0
GA 1000/10	874	888	4.1	1869	0
GA 1000/50	878	883	**2.5**	1305	0
GA 1000/100	865	878	5.6	4309	0
GA 10000/10	872	877	**2.5**	6999	0
GA 10000/50	875	879	3.1	5317	0
GA 10000/100	869	875	4.5	10733	0
LS 10s	771	826	31.7	**10**	0
LS 30s	740	791	47.0	30	0
LS 5m	**554**	**554**	**0.0**	300	1.0

Performance of stochastic search techniques on a "Concanavalin A-like lectins/glucanases", a 12-stranded β-sandwich template. SA is Simulated Annealing, GA is Genetic Algorithm, and LS is Local Search. For Simulated Annealing, we show results for convergence thresholds of 200, 500, and 1000 generations. For the Genetic Algorithm, we show results for convergence thresholds of 10, 50, and 100 generations, and for population sizes of 1000 and 10000. For Local Search, we show results for time limits of 10 seconds, 30 seconds and five minutes, on a 12-core AMD Opteron. The "Optimal" column indicates the fraction of runs for each search method that achieved the global optimum.

cessing cores. We measured the run-time performance of MRFy's genetic algorithm implementation (with a fixed random seed) on the "8-bladed β-propeller" template. The model has 343 nodes, of which 178 appear in 40 β-strands. The segments between β-strands typically have at most 10 nodes. We used a query sequence of 592 amino acids, but each placement breaks the sequence into 41 pieces, each of which typically has at most 20 amino acids. Because MRFy can solve the models between the β-strands independently, this benchmark has a lot of parallelism.

Figure 4.4 shows speedups when using from 1 to 48 of the cores on a 48-core, 2.3GHz AMD Opteron 6176 system. Errors are estimated from 5 runs. After about 12 cores, where MRFy runs 6 times as fast as sequential code, speedup rolls off. We note that by running 4 instances of MRFy in parallel on different searches, we would expect to be able to use all 48 cores with about 50% efficiency.

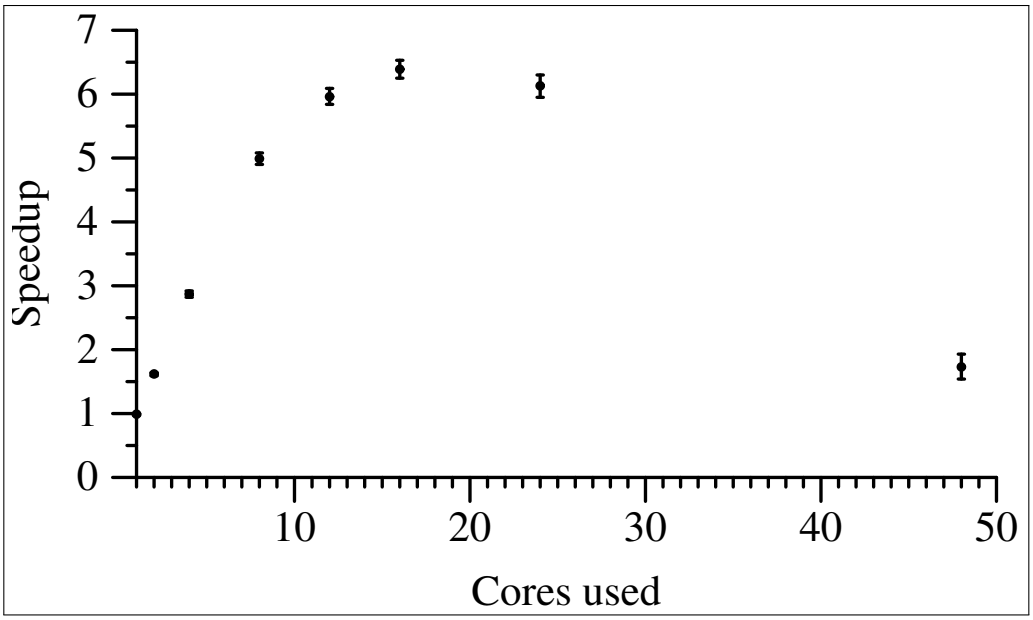


Figure 4.4: MRFy's parallel speedup on an 8-bladed β-propeller, using a 48-core system. After about 12 cores, speedup falls off.

4.3.2 Remote homology detection accuracy

We performed cross-validation testing on 11 β-barrel superfamilies, both with and without simulated evolution. For MRFy, the balance between accuracy and computational efficiency is determined by the termination conditions, as well as the search technique chosen. Because local search so dramatically outperformed simulated annealing and the genetic algorithm, we conducted these cross-validation tests only on local search. We chose 30 seconds as a balance between speed and accuracy; a 5 minute time limit might result in better accuracy, but for high-throughput, whole-genome scans, 5 minutes per alignment is excessive.

We compared MRFy's performance, both with and without simulated evolution, to the results from Chapter 3. Table 4.4 shows the area (AUC) under the Receiver Operator Characteristic (ROC) curve for MRFy, the very best result from SMURFLite, and HMMER, RAPTOR, and HHPred. Importantly, we are choosing the best SMURFLite parameters for each superfamily, which could not be known in advance; thus, we demonstrate improvements over the *very best* SMURFLite can perform, rather than just an average case.

We first note the "Barwin-like endoglucanases" superfamily highlighted in Chapter 3. SMURFLite performed better as the interleave threshold was increased on this superfamily, and also when simulated evolution was added. Since MRFy discards no β-strands, we were curious how it would perform on this superfamily. Notably, this superfamily has exceedingly little training data; during cross-validation, there are at most 4 training sequences and as few as 3 when filtered at a BLAST E-value of 10^{-7} and the family under test is left out. Without simulated evolution, MRFy achieves an AUC of 0.86, outperforming SMURFLite without simulated evolution (SMURFLite achieved an AUC of 0.77 with an interleave threshold of 2, and 0.81 with an interleave threshold of 4). When simulated evolution is added, MRFy achieves an AUC of 0.92, outperforming SMURFLite with an interleave threshold of 2, but falling just short of the 0.94 AUC SMURFLite demonstrates with an interleave threshold of 4 and simulated evolution.

MRFy outperforms SMURFLite in terms of AUC on four of the β-barrel superfamilies, while SMURFLite outperforms MRFy on three. There was only one superfamily, the "Prokaryotic SH3-related domain," where SMURFLite outperformed HMMER, RAPTOR, and HHPred while MRFy did not (MRFy, with an AUC of 0.73, fell behind HMMER's 0.81 AUC). Unfortunately, MRFy never produced the best performance on a superfamily that SMURFLite had not performed best on in our previous work. Thus, with the exception of the "Barwin-like endoglucanase" superfamily, the added β-strand information does not seem to help MRFy significantly in the cases where HMMER, RAPTOR, or HHPred performed best.

4.4 Discussion

We have presented MRFy, a method that uses stochastic search to find alignments of protein sequences to Markov random field models. MRFy in most cases outperforms SMURFLite, but we should consider several possible enhancements to MRFy that might improve its performance. As demonstrated on the β-sandwich superfamily,

Table 4.4: AUC on Beta-Barrel superfamilies

	HMMER	RAPTOR	HHPred	SMURFLite (best)	MRFy	MRFy, SE
MRFy performs best						
Translation proteins	-	-	0.66	0.93	**0.95**	0.91
Barwin-like endoglucanases	-	-	0.75	0.77	0.86	**0.92**
Tudor/PWWP/MBT	0.78	0.74	0.67	0.83	**0.86**	**0.86**
Nucleic acid-binding proteins	0.75	-	0.67	0.89	0.75	**0.95**
SMURFLite performs best						
Cyclophilin-like	0.67	0.61	0.7	**0.85**	0.82	0.80
Sm-like ribonucleoproteins	0.73	0.71	0.77	**0.85**	0.77	0.77
Prokaryotic SH3-related domain	0.81	-	-	**0.83**	0.73	0.72
HHPred performs best						
Translation proteins SH3-like	0.83	0.81	**0.86**	0.62	-	0.63
RAPTOR performs best						
PDZ domain-like	0.96	**1.0**	0.99	0.97	0.95	0.95
FMN-binding split barrel	0.62	**0.82**	0.61	-	-	-
HMMER performs best						
Electron Transport accessory proteins	**0.84**	-	0.77	0.66	-	0.68

Note: for SMURFLite, value indicated is the best of all values in Table 3.2. For MRFy, SimEv is simulated evolution. A dash ('-') in a result entry indicates the method failed on these structures, i.e. an AUC of less than 0.6

MRFy with local search achieves a globally optimal alignment when given 5 minutes of run-time, but fails to find a score close to this when given only 30 seconds. It was not immediately clear how to bring convergence testing into the local search model, but doing so might achieve results comparable to the 5 minute results in less time.

We hope that MRFy will be useful for whole-genome annotation of newly-sequenced organisms. The tradeoff of time versus accuracy suggests a two-phase approach to this task: a scan with relatively strict run-time performance requirements (perhaps no more than ten seconds per alignment) coupled with a relatively loose p-value threshold would produce a number of candidates, many of which would likely be false positives. Then, MRFy could be re-run on these candidates with more computationally demanding settings, and with a more strict p-value threshold. MRFy

computes p-values identically to SMURFLite: an extreme value distribution [Edd98] is fitted to a distribution of raw scores, and then a p-value is computed as $1 - cdf\,(x)$ for any raw MRFy score x. Computing the p-value accurately in the face of different search intensities might require fitting multiple distributions, each for a different level of search intensity. Otherwise, if the distribution is obtained with an intensive search, then at less-intensive search parameters, true positives may result in poor p-values; similarly, if the distribution is obtained with a quick search, then more-intensive search parameters might result in false positives scoring comparatively well, and appearing to have good p-values.

As in Chapter 3, we compared MRFy to HHPred [Söd05]. As discussed, HHPred has an advantage in that it builds profiles based on all of protein sequence space. As a future enhancement to MRFy, we plan to introduce query profiles, so that the MRFy alignment is to a sequence profile built from the query sequence, rather than just the query sequence. However, this will introduce a run-time performance hit in two ways. First, the time to run a sequence homology search using the BLAST [AMS$^+$97] family of tools can be significant, though the work on compressively-accelerated algorithms by Loh, et al.[LBB12] may reduce this impact. Second, computing the Viterbi and β-pairing scores naïvely will require time directly proportional to the number of sequences in the query profile. Representing these query sequences as sets of residue frequency vectors should help, and there may be other approaches to consider.

We have demonstrated that MRFy is an improvement to SMURFLite, one that brings the full power of a Markov random field to bear. Thus far, only β-strand interactions lead to non-local interactions in the MRFy Markov random field. In the future, we will investigate fitting other secondary structural elements (the α-helices) into this model. In addition, disulfide bonds, which can occur between cysteine residues and have been shown to be highly conserved [NAL09, TMC$^+$12], would appear to fit easily into this model.

Chapter 5

Conclusion and Future Work

5.1 Contrasting Markov random field approaches

We have explored two approaches to making the SMURF [Men09, MBC10] Markov random field model computationally tractable on all protein folds. SMURF used multidimensional dynamic programming to exactly compute the optimal energy function on a β-structural Markov random field, which was computationally intractable when β-strands were highly interleaved. In Chapter 3, we demonstrated a method, SMURFLite, for simplifying the Markov random field itself, by removing only those nonlocal interactions that caused the computational complexity to grow beyond reasonable bounds. In addition, SMURFLite uses "simulated evolution" to mitigate, at least in part, the information loss that this simplification poses.

In Chapter 4, we demonstrated an alternative method, MRFy, for approximating a solution to the full SMURF Markov random field, without discarding any β-strand information. This method, too, benefits from simulated evolution, though this benefit seems primarily confined to the protein superfamilies that have barely-adequate training data.

In essence, SMURFLite *exactly* computes the solution to an *approximation* of the SMURF Markov random field, while MRFy *approximately* computes a solution to the *exact* SMURF Markov random field.

A natural extension of this work would be to combine the methodologies

from Chapters 3 and 4. One approach to this would be to use SMURFLite at a low interleave threshold, such as 2, to produce an alignment that could serve as an initial guess for MRFy's placement of β-strand residues. Such an alignment would, at times, need to be modified to fit the β-strands that had been ignored by SMURFLite.

We also propose to evaluate this combined approach in comparison to the newer threading approach, RaptorX [PX11b], which incorporates multiple-template alignments and solvent-accessibility information.

5.2 Structurally consistent superfamilies

The results from Chapters 2 and 3 suggest that a purely structural basis for remote homology detection may result in, in some sense, an unfair test. Some SCOP superfamilies exhibit structural inconsistency; this, as well as historical artifacts such as "dustbin families" as suggested by [PLG12], pose a considerable challenge. In addition, the structural *consistency* of the β-propeller folds appears to be relatively unusual at the fold level. Given that the methods explored in this work rely upon high-quality structural alignments that, ideally, preserve highly-conserved secondary-structural regions, efforts to improve these alignments would be beneficial. Recent work in the field of structural alignment, including our work [DNC12] and that of Wang, et al. [Wan12] may preserve more sequence and secondary-structural similarity, occasionally at the expense of small amounts of structural alignment quality.

We may also consider the task of remote homology detection when freed from the occasional inconsistencies of SCOP; evaluating SMURFLite and MRFy on a purely structurally-derived hierarchy, as described in Chapter 2, may be worth exploring.

5.3 MRFy with sequence profiles

MRFy, along with SMURFLite, relies on computing an alignment of a single protein sequence to a Markov random field. As has been demonstrated by approaches such as PSI-BLAST [AMS+97] and BetaWrapPro [MMP+05], incorporating homologous profile data can improve both close and remote homology detection. Given MRFy's stochastic search approach, we expect that the less-discretized residue composition of each column of a query profile, as compared with a single query sequence, would smooth out the fitness landscape of MRFy's objective function (namely, the SMURF energy function) and thus enable the local-improvement feature of MRFy's local search to be more efficient. As discussed in Chapter 4, two computational challenges arise: how to quickly find close sequence homologs to build a profile, and how to quickly score a profile against a Markov random field. The work of Loh, et al. [LBB12], as well as our recent but as-yet unpublished work on compressively-accelerated genomic and protein sequence search algorithms, provides a partial solution to the first performance concern.

Regarding the second performance concern, how to quickly score a profile against a Markov random field, we note that HHPred [Söd05] and its relative, HHBlits [RBHS12] perform HMM-HMM alignment. They solve the slightly simpler problem of aligning a hidden Markov model built from a *query* profile with a hidden Markov model built from training data. This raises the question: could we perform MRF-MRF alignment, or at least HMM-MRF alignment? Our current model of Markov random fields requires a structural alignment in order to annotate β-strands, but a hidden Markov model can be built from a sequence alignment. Could we build a hidden Markov model from a query sequence (and resulting profile), and align it to our Markov random field model? This appears to bear further consideration.

5.4 Extension to Other Protein Classes

Both SMURFLite and MRFy differ from hidden Markov models, such as those employed by HMMER [Edd98], only in that they incorporate non-local interactions between residues participating in hydrogen bonds between β-strands. This means that SMURFLite and MRFy are most at home in the SCOP class of "all beta proteins," and while we may also be able to show benefits in the classes of "alpha/beta" and "alpha+beta" proteins, we would expect to contribute little to, and in fact overtrain on, the "all alpha proteins," given that there are occasional β-strands in the α-helical proteins, just as there are the converse (for example, the "Barwin" protein discussed in Chapters 3 and 4 has four α-helices in addition to its eight β-strands).

We intend to extend MRFy to incorporate α-helix conditional probabilities, as explored by Cao, et al. [CC12] within the context of HMMER. At the simplest level, highly-conserved α-helices in between β-strands should prevent those β-strands from occluding the helices; incorporating the α-helical residue propensities is an obvious extension of the model.

5.5 More Generalized Contact Maps

Beyond specific secondary-structural elements, evolution conserves other non-local interactions. Disulfide bonds, which occur between cysteine residues in proteins, anchor certain protein structures, and are thus highly conserved. MRFy's model of non-local interactions could easily incorporate these pairwise bonds; the probability of a disulfide bond between two cysteine residues would be close to 1, while the probability for any other pairing would be zero, or close to zero. Other, less common interactions such as peroxide and diselenide bonds may also lend themselves to this model.

Beyond specific chemical bonds, we may consider any structural core that appears to be highly conserved within a homologous group of proteins to be a candidate for our model of a Markov random field, encompassing non-local interactions. The main prerequisite for such an extension would be the availability of adequate

training data to build a model of conditional probabilities for the non-local interactions in question.

Bibliography

[AGM+90] S F Altschul, W Gish, W Miller, E W Myers, and D J Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990.

[AHB+04] Antonina Andreeva, Dave Howorth, Steven E Brenner, Tim J P Hubbard, Cyrus Chothia, and Alexey G Murzin. SCOP database in 2004: refinements integrate structure and sequence family data. *Nucleic Acids Research*, 32(Database issue):D226–9, January 2004.

[AMS+97] S F Altschul, T L Madden, A A Schäffer, J Zhang, Z Zhang, W Miller, and D J Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, September 1997.

[BAD03] D Beck, R Armen, and V Daggett. A consensus view of fold space: Combining SCOP, CATH, and the Dali Domain Dictionary. *Protein Science*, 2003.

[BBB+00] H M Berman, T N Bhat, P E Bourne, Z Feng, G Gilliland, H Weissig, and J Westbrook. The Protein Data Bank and the challenge of structural genomics. *Nature structural biology*, 7 Suppl:957–959, November 2000.

[BBC99] E Bornberg-Bauer and H S Chan. Modeling evolutionary landscapes: mutational stability, topology, and superfunnels in sequence space. In *Proceedings of the National ...*, 1999.

[BCHM94] P Baldi, Y Chauvin, T Hunkapiller, and M A McClure. Hidden Markov models of biological primary sequence information. *Proceedings of the National Academy of Science*, 91(3):1059–1063, February 1994.

[BCM+01] P Bradley, L Cowen, M Menke, J King, and B Berger. BETAWRAP: successful prediction of parallel -helices from primary sequence reveals an association with many microbial pathogens. *Proceedings of the National Academy of Science*, 98(26):14819–14824, 2001.

[BKW+77] F C Bernstein, T F Koetzle, G J Williams, EE Meyer, M D Brice, J R Rodgers, O Kennard, T Shimanouchi, and M Tasumi. The Protein Data Bank: a computer-based archival file for macromolecular structures. *Journal of Molecular Biology*, 112:535, 1977.

[BL98] B Berger and T Leighton. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *Journal of Computational Biology*, 1998.

[BPSW70] L E Baum, T Petrie, G Soules, and N Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The annals of mathematical statistics*, 1970.

[BSA+12] Grzegorz M Boratyn, Alejandro A Schäffer, Richa Agarwala, Stephen F Altschul, David J Lipman, and Thomas L Madden. Domain enhanced lookup time accelerated BLAST. *Biology direct*, 7:12, 2012.

[BSL09] C Berbalk, C S Schwaiger, and P Lackner. Accuracy analysis of multiple structure alignments. *Protein Science*, 2009.

[CBM+02] L Cowen, P Bradley, M Menke, J King, and B Berger. Predicting the beta-helix fold from protein sequence data. *Journal of Computational Biology*, 9(2):261–276, 2002.

[CC12] Mengfei Cao and Lenore J Cowen. Remote homology detection on alpha-structural proteins using simulated evolution. In *BCB '12: Pro-*

ceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine. ACM Request Permissions, October 2012.

[Chu89] G A Churchill. Stochastic models for heterogeneous DNA sequences. *Bulletin of mathematical biology*, 1989.

[CK06] In-Geol Choi and Sung-Hou Kim. Evolution of protein structural classes and protein sequence families. *Proceedings of the National Academy of Science*, 103(38):14056–14061, September 2006.

[CL86] C Chothia and A M Lesk. The relation between the divergence of sequence and structure in proteins. *The EMBO journal*, 5(4):823–826, April 1986.

[CQKK04] S Cheek, Y Qi, S S Krishna, and L N Kinch. SCOPmap: automated assignment of protein structures to evolutionary superfamilies. *BMC Bioinformatics*, 2004.

[CSL+09] A L Cuff, I Sillitoe, T Lewis, O C Redfern, R Garratt, J Thornton, and C A Orengo. The CATH classification revisited–architectures reviewed and new ways to characterize structural divergence in superfamilies. *Nucleic Acids Research*, 37(Database):D310–D314, January 2009.

[CSX06] Pin-Hao Chi, Chi-Ren Shyu, and Dong Xu. A fast SCOP fold classification system using content-based E-Predict algorithm. *BMC Bioinformatics*, 7:362, 2006.

[Dav91] L Davis. Bit-climbing, representational bias, and test suite design. In R K Belew and L B Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 18–23, San Mateo, CA, 1991.

[DBR97] S Dalal, S Balasubramanian, and L Regan. Protein alchemy: changing beta-sheet into alpha-helix. *Nature structural biology*, 4(7):548–552, July 1997.

[DC97] K A Dill and H S Chan. From Levinthal to pathways to funnels. *Nature structural biology*, 4(1):10–19, January 1997.

[DGR12] Noah M Daniels, Andrew Gallant, and Norman Ramsey. Experience report: Haskell in computational biology. In *Proceedings of the 17th ACM ....* ACM Request Permissions, September 2012.

[DHBC12] Noah M Daniels, Raghavendra Hosur, Bonnie Berger, and Lenore J Cowen. SMURFLite: combining simplified Markov random fields with simulated evolution improves remote homology detection for beta-structural proteins into the twilight zone. *Bioinformatics*, 28(9):1216–1222, May 2012.

[DKCM11] Noah Daniels, Anoop Kumar, Lenore Cowen, and Matt Menke. Touring Protein Space with Matt. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, April 2011.

[DLC02] N Dasgupta, S Lin, and L Carin. Sequential modeling for identifying CpG island locations in human genome. *Signal Processing Letters*, 2002.

[DNC12] Noah M Daniels, Shilpa Nadimpalli, and Lenore J Cowen. Formatt: Correcting protein multiple structural alignments by incorporating sequence alignment. *BMC Bioinformatics*, 13(1):259, 2012.

[DSSD00] A R Dinner, A Sali, L J Smith, and C M Dobson. Understanding protein folding via free-energy surfaces from theory and experiment. *Trends in Biochemical Sciences*, 2000.

[Dun06] Roland L Dunbrack, Jr. Sequence comparison and protein structure prediction. *Current Opinion in Structural Biology*, 16(3):374–384, June 2006.

[Edd98] S R Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9):755–763, October 1998.

[ES99] A Elofsson and E L Sonnhammer. A comparison of sequence and structure protein domain families as a basis for structural genomics. *Bioinformatics*, 15(6):480–500, June 1999.

[FMSB+06] Robert D Finn, Jaina Mistry, Benjamin Schuster-Böckler, Sam Griffiths-Jones, Volker Hollich, Timo Lassmann, Simon Moxon, Mhairi Marshall, Ajay Khanna, Richard Durbin, Sean R Eddy, Erik L L Sonnhammer, and Alex Bateman. Pfam: clans, web tools and services. *Nucleic Acids Research*, 34(Database issue):D247–51, January 2006.

[FTM+08] Robert D Finn, John Tate, Jaina Mistry, Penny C Coggill, Stephen John Sammut, Hans-Rudolf Hotz, Goran Ceric, Kristoffer Forslund, Sean R Eddy, Erik L L Sonnhammer, and Alex Bateman. The Pfam protein families database. *Nucleic Acids Research*, 36(Database issue):D281–8, January 2008.

[GL98] M Gerstein and M Levitt. Comprehensive assessment of automatic structural alignment against a manual standard, the Scop Classification of Proteins. *Protein Science*, 1998.

[GLA+07] Lesley H Greene, Tony E Lewis, Sarah Addou, Alison Cuff, Tim Dallman, Mark Dibley, Oliver Redfern, Frances Pearl, Rekha Nambudiry, Adam Reid, Ian Sillitoe, Corin Yeats, Janet M Thornton, and Christine A Orengo. The CATH domain structure database: new protocols and classification levels give a more comprehensive resource for exploring evolution. *Nucleic Acids Research*, 35(Database issue):D291–7, January 2007.

[GMB96] J F Gibrat, T Madej, and S H Bryant. Surprising similarities in structure comparison. *Current Opinion in Structural Biology*, 6(3):377–385, June 1996.

[GVSD02] Gad Getz, Michele Vendruscolo, David Sachs, and Eytan Domany. Automated assignment of SCOP and CATH protein structure classifications from FSSP scores. *Proteins: Structure, Function, and Bioinformatics*, 46(4):405–415, March 2002.

[GW09] V Gopalakrishnan and P Weigele. Conditional graphical models for protein structural motif recognition. *Journal of Computational Biology*, 2009.

[HBB+06] Nicolas Hulo, Amos Bairoch, Virginie Bulliard, Lorenzo Cerutti, Edouard De Castro, Petra S Langendijk-Genevaux, Marco Pagni, and Christian J A Sigrist. The PROSITE database. *Nucleic Acids Research*, 34(Database issue):D227–30, January 2006.

[Heu99] V Heun. Approximate protein folding in the HP side chain model on extended cubic lattices. *Algorithms-ESA'99*, 1999.

[HH92] S Henikoff and J G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Science*, 89(22):10915–10919, November 1992.

[HJ99] C Hadley and D Jones. A systematic comparison of protein structure classifications: SCOP, CATH and FSSP. *Structure*, 1999.

[HK96] R Hughey and A Krogh. Hidden Markov models for sequence analysis: extension and analysis of the basic method. *Computer applications in the biosciences : CABIOS*, 12(2):95–107, April 1996.

[HLKT86] R Huber, T A Langworthy, H König, and M Thomm. *Thermotoga maritima* sp. nov. represents a new genus of unique extremely thermophilic eubacteria growing up to 90 C. *Archives of Microbiology*, 144:324–333, 1986.

[HM05] P V Hentenryck and L Michel. *Constraint-based local search.* The MIT Press, Cambridge, MA, 2005.

[HP00] L Holm and J Park. DaliLite workbench for protein structure comparison. *Bioinformatics*, 16(6):566–567, June 2000.

[HPM+02] Andrew Harrison, Frances Pearl, Richard Mott, Janet Thornton, and Christine Orengo. Quantifying the similarities within fold space. *Journal of Molecular Biology*, 323(5):909–926, November 2002.

[HR77] John H Holland and Judith S Reitman. Cognitive systems based on adaptive algorithms. *ACM SIGART Bulletin*, (63):49, June 1977.

[HS96] L Holm and C Sander. The FSSP database: fold classification based on structure-structure alignment of proteins. *Nucleic Acids Research*, 24(1):206–209, January 1996.

[HS98] L Holm and C Sander. Touring protein fold space with Dali/FSSP. *Nucleic Acids Research*, 26(1):316–319, January 1998.

[HVS06] T A Holland, S Veretnik, and I N Shindyalov. Partitioning protein structures into domains: why is it so difficult? *Journal of Molecular Biology*, 2006.

[Joh06] Stephen Johnson. Remote Protein Homology Detection Using Hidden Markov Models. *PhD thesis, Washington University in St. Louis*, pages 1–106, October 2006.

[Jon97] D T Jones. Progress in protein structure prediction. *Current Opinion in Structural Biology*, 7(3):377–387, June 1997.

[Jon99] D T Jones. GenTHREADER: an efficient and reliable protein fold recognition method for genomic sequences. *Journal of Molecular Biology*, 287(4):797–815, April 1999.

[JTT92] D T Jones, W R Taylort, and J M Thornton. A new approach to protein fold recognition. *Nature*, 1992.

[JVP06] G Jayachandran, V Vishal, and V S Pande. Using massively parallel simulation and Markovian models to study protein folding: Examining the dynamics of the villin headpiece. *The Journal of chemical physics*, 2006.

[Kab76] W Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics*, 1976.

[KBH98] K Karplus, C Barrett, and Richard Hughey. Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, 14:846–856, 1998.

[KBM+94] Anders Krogh, Michael Brown, I Saira Mian, Kimmen Sjölander, and David Haussler. Hidden Markov Models in Computational Biology. *Journal of Molecular Biology*, 235(5):1501–1531, February 1994.

[KC09] Anoop Kumar and Lenore Cowen. Augmented training of hidden Markov models to recognize remote homologs via simulated evolution. *Bioinformatics*, 25(13):1602–1608, 2009.

[KC10] Anoop Kumar and Lenore Cowen. Recognition of beta-structural motifs using hidden Markov models trained with simulated evolution. *Bioinformatics*, 26(ISMB 2010):i287–i293, 2010.

[KKL05] R Kolodny, P Koehl, and M Levitt. Comprehensive evaluation of protein structure alignment methods: scoring by geometric measures. *Journal of Molecular Biology*, 2005.

[KPH06] Rachel Kolodny, Donald Petrey, and Barry Honig. Protein structure comparison: implications for the nature of 'fold space', and structure and function prediction. *Current Opinion in Structural Biology*, 16(3):393–398, June 2006.

[KV83] S Kirkpatrick and M P Vecchi. Optimization by simmulated annealing. *Science (New York, N.Y.)*, 1983.

[LBB12] Po-Ru Loh, Michael Baym, and Bonnie Berger. Compressive genomics. *Nature biotechnology*, 30(7):627–630, July 2012.

[Lev69] C Levinthal. How to Fold Gracefully. *Mossbauer Spectroscopy in Biological Systems: Proceedings of a meeting held at Allerton House, Monticello, Illinois*, pages 22–24, 1969.

[LS80] S Lifson and C Sander. Specific recognition in the tertiary structure of [beta]-sheets of proteins. *Journal of Molecular Biology*, 139:627–629, 1980.

[LS96] R H Lathrop and T F Smith. Global optimum protein threading with gapped alignment and empirical pair score functions. *Journal of Molecular Biology*, 255(4):641–665, February 1996.

[MBA05] A Marchler-Bauer and J B Anderson. CDD: a Conserved Domain Database for protein classification. *Nucleic Acids Research*, 2005.

[MBB95] T Madej, M S Boguski, and S H Bryant. Threading analysis suggests that the obese gene product may be a helical cytokine. *FEBS letters*, 373(1):13–18, October 1995.

[MBC08] M Menke, B Berger, and L Cowen. Matt: local flexibility aids protein multiple structure alignment. *PLoS Computational Biology*, 2008.

[MBC10] M Menke, B Berger, and L Cowen. Markov random fields reveal an N-terminal double beta-propeller motif as part of a bacterial hybrid two-component sensor system. *Proceedings of the National Academy of Science*, 2010.

[MBH95] A Murzin, S Brenner, and T Hubbard. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247(4):536–540, 1995.

[MBJ00] L J McGuffin, K Bryson, and D T Jones. The PSIPRED protein structure prediction server. *Bioinformatics*, 2000.

[MCBR98] A Matagne, E W Chung, L J Ball, and S E Radford. The origin of the -domain intermediate in the folding of hen lysozyme. *Journal of Molecular Biology*, 1998.

[MDBO98] K Mizuguchi, C M Deane, T L Blundell, and J P Overington. HOMSTRAD: a database of protein structure alignments for homologous families. *Protein Science*, 7(11):2469–2471, November 1998.

[Men09] M Menke. Computational approaches to modeling the conserved structural core among distantly homolgous proteins. *Massachusetts. PhD Thesis, MIT*, 2009.

[MMP+05] A McDonnell, M Menke, N Palmer, J King, and L Cowen. Comparative modeling of mainly-beta proteins by profile wrapping. *broad.mit.edu*, 2005.

[Mou06] J Moult. Rigorous performance evaluation in protein structure modelling and implications for computational biology. *Transactions of the Royal Society: Biological Sciences*, 2006.

[MSE96] M A McClure, C Smith, and P Elton. Parameterization studies for the SAM and HMMER methods of hidden Markov model generation., 1996.

[NAL09] Guy Naamati, Manor Askenazi, and Michal Linial. ClanTox: a classifier of short animal toxins. *Nucleic Acids Research*, 37(Web Server issue):W363–8, July 2009.

[OJT94] C A Orengo, D T Jones, and J M Thornton. Protein superfamilies and domain superfolds. *Nature*, 372(6507):631–634, December 1994.

[OMJ+97] C A Orengo, A D Michie, S Jones, D T Jones, M B Swindells, and J M Thornton. CATH–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1108, August 1997.

[ORV99] O Olmea, Burkhard Rost, and Alfonso Valencia. Effective use of sequence correlation and conservation in fold recognition. *Journal of Molecular Biology*, 293(5):1221–1239, 1999.

[OSO09] A R Ortiz, CEM Strauss, and O Olmea. MAMMOTH (matching molecular models obtained from theory): an automated method for model comparison. *Protein Science*, 2009.

[PBB+03] F M G Pearl, C F Bennett, J E Bray, A P Harrison, N Martin, A Shepherd, I Sillitoe, J Thornton, and C A Orengo. The CATH database: an extended protein family resource for structural and functional genomics. *Nucleic Acids Research*, 31(1):452–455, January 2003.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Networks of Plausble Inference. Morgan Kaufmann Pub, 1988.

[PLG12] Ralph B Pethica, Michael Levitt, and Julian Gough. Evolutionarily consistent families in SCOP: sequence, structure and function. *BMC Structural Biology*, 12(1):27, 2012.

[PX11a] J Peng and J Xu. A multipletemplate approach to protein threading. *Proteins: Structure, Function, and Bioinformatics*, 2011.

[PX11b] J Peng and J Xu. RaptorX: exploiting structure information for protein alignment by statistical inference. *Proteins: Structure, Function, and Bioinformatics*, 2011.

[Rab89] L R Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, 1989.

[RBHS12] Michael Remmert, Andreas Biegert, Andreas Hauser, and Johannes Söding. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nature methods*, 9(2):173–175, February 2012.

[RHD07] O Redfern, A Harrison, and T Dallman. CATHEDRAL: a fast and effective algorithm to predict folds and domain boundaries from multidomain protein structures. *PLoS Computational Biology*, 2007.

[Ros99] B Rost. Twilight zone of protein sequence alignments. *Protein Engineering*, 12(2):85–94, February 1999.

[Ros02] B Rost. Did evolution leap to create the protein universe? *Current Opinion in Structural Biology*, 2002.

[RSWD09] Jairo Rocha, Joan Segura, Richard C Wilson, and Swagata Dasgupta. Flexible structural protein alignment by a sequence of local transformations. *Bioinformatics*, 25(13):1625–1631, July 2009.

[SB98] I N Shindyalov and P E Bourne. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Engineering*, 1998.

[SB00] I N Shindyalov and P E Bourne. An alternative view of protein fold space. *Proteins: Structure, Function, and Bioinformatics*, 2000.

[SBL05] Johannes Söding, Andreas Biegert, and Andrei N Lupas. The HHpred interactive server for protein homology detection and structure prediction. *Nucleic Acids Research*, 33(Web Server issue):W244–8, July 2005.

[SDD+07] David E Shaw, Martin M Deneroff, Ron O Dror, Jeffrey S Kuskin, Richard H Larson, John K Salmon, Cliff Young, Brannon Batson, Kevin J Bowers, Jack C Chao, Michael P Eastwood, Joseph Gagliardo, J P Grossman, C Richard Ho, Douglas J Ierardi, István Kolossváry, John L Klepeis, Timothy Layman, Christine McLeavey, Mark A Moraes, Rolf Mueller, Edward C Priest, Yibing Shan, Jochen Spengler, Michael Theobald, Brian Towles, and Stanley C Wang. Anton, a special-purpose machine for molecular dynamics simulation. In *ISCA*

'07: Proceedings of the 34th annual international symposium on Computer architecture. ACM Request Permissions, June 2007.

[SHJ97] P Smyth, D Heckerman, and M I Jordan. Probabilistic independence networks for hidden Markov probability models. *Neural computation*, 9(2):227–269, February 1997.

[SKG09] Ruslan I Sadreyev, Bong-Hyun Kim, and Nick V Grishin. Discrete-continuous duality of protein structure space. *Current Opinion in Structural Biology*, 19(3):321–328, June 2009.

[SKP08] Paolo Sonego, András Kocsor, and Sándor Pongor. ROC analysis: applications to the classification of biological sequences and 3D structures. *Briefings in bioinformatics*, 9(3):198–209, May 2008.

[SMP08] M Simonsen, T Mailund, and CNS Pedersen. Rapid Neighbour-Joining. *Lect Notes Comput Sci*, 5251:113:122, 2008.

[SMW95] RA Sayle and EJ Milner-White. RASMOL: biomolecular graphics for all. *Trends in Biochemical Sciences*, 20(9):374, 1995.

[Söd05] Johannes Söding. Protein homology detection by HMM-HMM comparison. *Bioinformatics*, 21(7):951–960, March 2005.

[SSH+92] Birte Svensson, Ib Svendsen, Peter Hoejrup, Peter Roepstorff, Svend Ludvigsen, and Flemming M Poulsen. Primary structure of barwin: a barley seed protein closely related to the C-terminal domain of proteins encoded by wound-induced plant genes. *Biochemistry*, 1992.

[ST02] R E Steward and J M Thorton. Prediction of strand pairing in antiparallel and parallel -sheets using information theory. *Proteins: Structure, Function, and Bioinformatics*, 48:178–191, 2002.

[STG+06] V Sam, C H Tai, J Garnier, J F Gibrat, and B Lee. ROC and confusion analysis of structure comparison methods identify the main causes of

divergence from manual protein classification. *BMC Bioinformatics*, 2006.

[SW81] T F Smith and M S Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, March 1981.

[SWS07] Stefan J Suhrer, Markus Wiederstein, and Manfred J Sippl. QSCOP–SCOP quantified by structural relationships. *Bioinformatics*, 23(4):513–514, February 2007.

[TGG+08] C Tai, J Garnier, J Gibrat, B Lee, and P Munson. Towards an automatic classification of protein structural domains based on .... *BMC Bioinformatics*, 2008.

[TKMN99] C J Tsai, S Kumar, B Ma, and R Nussinov. Folding funnels, binding funnels, and protein function. *Protein Science*, 1999.

[TMC+12] Y Tirosh, N Morpurgo, M Cohen, M Linial, and G Bloch. Raalin, a transcript enriched in the honey bee brain, is a remnant of genomic rearrangement in *Hymenoptera*. *Insect molecular biology*, 21(3):305–318, June 2012.

[TRBK08] J Thomas, N Ramakrishnan, and C Bailey-Kellogg. Graphical models of residue coupling in protein families. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, 2008.

[VBAS04] Stella Veretnik, Philip E Bourne, Nickolai N Alexandrov, and Ilya N Shindyalov. Toward consistent assignment of structural domains in proteins. *Journal of Molecular Biology*, 339(3):647–678, June 2004.

[VC06] M Vuk and T Curk. ROC curve, lift chart and calibration plot. *Metodoloski zvezki*, 2006.

[Vit67] A Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.

[VWLW05] I Van Walle, I Lasters, and L Wyns. SABmark—a benchmark for sequence alignment that covers the entire known fold space. *Bioinformatics*, 2005.

[VYB09] R Valas, S Yang, and P Bourne. Nothing about protein structure classification makes sense except in the light of evolution. *Current Opinion in Structural Biology*, 2009.

[Wan12] Shen Wang. *Protein structure alignment beyond spatial proximity*. In *Protein structure alignment beyond spatial proximity*, Long Beach, CA, July 2012.

[WJ94] LUSHENG WANG and TAO JIANG. On the Complexity of Multiple Sequence Alignment. *Journal of Computational Biology*, 1(4):337–348, January 1994.

[WJ99] Y Weiss and MI Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of Uncertainty in AI*, 1999.

[WKG00] C A Wilson, J Kreychman, and M Gerstein. Assessing annotation transfer for genomics: quantifying the relations between protein sequence, structure and function through traditional and probabilistic scores. *Journal of Molecular Biology*, 297(1):233–249, March 2000.

[WMS94] J V White, I Muchnik, and T F Smith. Modeling protein cores with Markov random fields. *Mathematical biosciences*, 124(2):149–179, December 1994.

[WMV+07] Derek Wilson, Martin Madera, Christine Vogel, Cyrus Chothia, and Julian Gough. The SUPERFAMILY database in 2007: families and

functions. *Nucleic Acids Research*, 35(Database issue):D308–13, January 2007.

[WS04] Markus Wistrand and Erik L L Sonnhammer. Improving profile HMM discrimination by adapting transition probabilities. *Journal of Molecular Biology*, 338(4):847–854, May 2004.

[WZ08] S Wu and Y Zhang. MUSTER: improving protein sequence profile–profile alignments by using multiple sources of structure information. *Proteins: Structure, Function, and Bioinformatics*, 2008.

[XLKX03] Jinbo Xu, Ming Li, Dongsup Kim, and Ying Xu. Raptor: Optimal Protein Threading By Linear Programming. *Journal of Bioinformatics and Computational Biology*, 1(1):95–117, 2003.

[YFZZ11] Yuedong Yang, Eshel Faraggi, Huiying Zhao, and Yaoqi Zhou. Improving protein fold recognition and template-based modeling by employing probabilistic-based matching between predicted one-dimensional structural properties of query and corresponding native properties of templates. *Bioinformatics*, 27(15):2076–2082, August 2011.

[ZB99] H Zhu and W Braun. Sequence specificity, statistical potentials, and three-dimensional structure prediction with self-correcting distance geometry calculations of beta-sheet formation in proteins. *Protein Science*, 8(2):326–342, 1999.

[ZGS+07] Adam Zemla, Brian Geisbrecht, Jason Smith, Marisa Lam, Bonnie Kirkpatrick, Mark Wagner, Tom Slezak, and Carol Ecale Zhou. STRALCP–structure alignment-based clustering of proteins. *Nucleic Acids Research*, 35(22):e150, 2007.

[ZS05] Yang Zhang and Jeffrey Skolnick. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Research*, 33(7):2302–2309, 2005.

[ZTW+09] Ying Zhang, Ines Thiele, Dana Weekes, Zhanwen Li, Lukasz Jaroszewski, Krzysztof Ginalski, Ashley M Deacon, John Wooley, Scott A Lesley, Ian A Wilson, Bernhard Palsson, Andrei Osterman, and Adam Godzik. Three-dimensional structural view of the central metabolic network of Thermotoga maritima. *Science (New York, N.Y.)*, 325(5947):1544–1549, September 2009.

Appendices

Pairwise scores for β-structural proteins

Table 1: Pairwise scores (negative log of probability) for buried β-strands

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y	X
A	2.84	2.89	2.41	1.63	2.58	3.31	2.06	2.54	2.89	2.42	2.91	2.30	2.66	3.06	3.52	2.68	2.47	2.54	2.53	2.60	9.21
C	3.77	2.19	3.33	3.71	3.56	3.13	3.44	3.65	2.89	3.57	4.20	3.40	1.96	3.76	3.52	3.14	3.16	3.81	2.78	3.47	9.21
D	4.56	4.59	4.73	4.73	5.06	3.82	4.14	4.78	4.73	5.55	4.61	2.99	3.06	4.73	2.83	3.47	5.11	4.73	4.73	5.78	9.21
E	4.09	5.28	5.04	3.02	4.66	5.61	3.44	5.03	2.19	5.32	4.20	5.04	5.04	5.04	5.04	4.39	5.11	6.52	5.04	5.08	9.21
F	2.30	2.39	2.63	1.92	2.12	1.97	2.19	2.47	2.19	2.39	2.13	2.99	2.15	1.96	2.42	1.95	2.16	2.40	2.42	2.19	9.21
G	3.87	2.80	2.23	3.71	2.81	2.72	2.53	3.32	3.14	3.29	3.51	4.09	1.96	3.76	3.52	3.47	2.80	3.10	3.62	2.94	9.21
H	4.09	4.59	4.02	3.02	4.50	4.00	2.75	5.03	4.61	4.99	4.61	4.09	4.61	3.76	4.61	3.29	4.41	5.82	4.72	4.68	9.21
I	1.73	1.95	1.82	1.76	1.94	1.95	2.19	1.58	1.79	1.72	1.67	2.14	2.37	1.96	2.83	1.86	2.47	1.73	1.83	1.84	9.21
K	6.17	5.28	5.87	3.02	5.76	5.87	5.87	5.88	5.87	5.32	5.87	5.87	5.87	3.76	3.52	5.08	5.87	6.11	5.87	5.87	9.21
L	1.66	1.92	2.63	2.10	1.91	1.97	2.19	1.77	1.28	1.71	1.78	2.01	2.37	2.15	2.83	1.99	1.89	1.83	2.01	1.79	9.21
M	3.77	4.18	3.33	2.61	3.27	3.82	3.44	3.34	3.45	3.41	2.82	4.09	3.76	2.37	3.52	5.08	3.16	3.52	3.62	3.38	9.21
N	4.38	4.59	2.92	4.66	5.35	5.61	4.14	5.03	4.66	4.85	5.30	4.66	3.76	3.06	3.52	4.39	3.32	4.91	3.11	5.78	9.21
P	5.07	3.49	3.33	5.00	4.84	3.82	5.00	5.59	5.00	5.55	5.30	4.09	5.00	3.76	5.00	4.39	5.11	6.11	3.34	5.08	9.21
Q	5.48	5.28	5.00	5.00	4.66	5.61	4.14	5.19	2.89	5.32	3.92	3.40	3.76	5.00	3.52	3.98	5.11	5.01	4.03	5.78	9.21
R	6.17	5.28	3.33	5.23	5.35	5.61	5.23	6.29	2.89	6.24	5.30	4.09	5.23	3.76	5.23	5.08	3.72	5.01	4.03	4.17	9.21
S	3.77	3.34	2.41	3.02	3.31	4.00	2.35	3.76	2.89	3.84	5.30	3.40	3.06	2.66	3.52	2.44	5.11	4.12	3.11	3.83	9.21
T	3.53	3.34	4.02	3.71	3.51	3.31	3.44	4.34	3.64	3.71	3.36	2.30	3.76	3.76	2.14	5.08	3.72	3.47	4.03	3.38	9.21
V	1.50	1.88	1.54	3.02	1.64	1.50	2.75	1.50	1.79	1.55	1.61	1.79	2.66	1.56	1.32	1.99	1.37	1.36	2.08	1.62	9.21
W	3.97	3.34	4.03	4.03	4.15	4.51	4.14	4.09	4.03	4.22	4.20	2.48	2.37	3.06	2.83	3.47	4.41	4.57	3.34	3.58	9.21
Y	2.99	2.98	4.02	3.02	2.87	2.78	3.04	3.05	2.98	2.94	2.91	4.09	3.06	3.76	1.91	3.14	2.71	3.05	2.53	3.00	9.21
X	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21

Table 2: Pairwise scores (negative log of probability) for exposed β-strands

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y	X
A	2.91	2.56	3.19	3.50	2.97	3.19	3.04	2.98	3.56	2.88	3.11	3.31	3.44	3.86	3.18	3.37	3.60	2.90	3.81	2.83	9.21
C	3.76	2.27	5.49	6.14	3.67	5.14	4.65	4.73	4.66	4.35	4.72	4.56	4.14	5.65	4.13	4.22	4.41	4.51	3.81	3.98	9.21
D	3.25	4.35	3.70	3.84	3.67	3.19	2.95	3.34	2.68	3.43	3.34	3.31	4.14	3.01	2.74	2.92	3.27	3.82	3.12	3.57	9.21
E	2.91	4.35	3.19	2.96	2.57	3.19	2.86	2.86	1.84	2.74	2.24	2.37	2.53	2.61	2.03	2.97	2.52	2.80	2.56	3.06	9.21
F	3.06	2.56	3.70	3.25	2.69	2.57	2.64	3.72	3.63	3.19	2.64	3.65	3.44	3.57	4.46	2.92	3.31	3.53	3.12	3.28	9.21
G	3.60	4.35	3.55	4.20	2.89	3.34	3.15	3.48	4.88	3.19	2.78	3.18	4.14	4.04	3.51	3.53	4.00	3.90	3.41	3.28	9.21
H	3.25	3.66	3.09	3.66	2.75	2.94	3.56	3.55	3.50	3.50	3.11	3.87	3.04	3.57	3.77	3.37	3.40	3.36	3.81	3.00	9.21
I	2.41	2.97	2.72	2.88	3.06	2.50	2.78	2.02	2.66	2.35	2.32	2.96	2.75	2.71	2.67	2.64	2.90	2.56	2.56	2.83	9.21
K	2.84	2.74	1.91	1.71	2.82	3.75	2.57	2.50	2.44	2.56	2.24	2.42	3.44	2.36	2.81	2.92	2.62	2.72	2.31	2.13	9.21
L	2.29	2.56	2.78	2.74	2.51	2.19	2.71	2.33	2.68	2.27	2.93	3.47	1.94	2.76	2.94	2.72	3.02	2.21	2.56	2.73	9.21
M	3.94	4.35	4.11	3.66	3.38	3.19	3.74	3.72	3.78	4.35	3.34	4.16	4.14	4.26	3.88	4.22	4.54	4.22	4.51	5.08	9.21
N	3.60	3.66	3.55	3.25	3.85	3.06	3.96	3.81	3.43	4.35	3.62	3.47	3.44	3.09	3.08	3.45	3.27	4.00	3.12	3.20	9.21
P	4.85	4.35	5.49	4.53	4.77	5.14	4.25	4.73	5.57	3.94	4.72	4.56	4.59	4.96	4.46	5.32	4.88	4.36	3.81	3.69	9.21
Q	3.76	4.35	2.85	3.10	3.38	3.53	3.27	3.17	2.97	3.25	3.34	2.69	3.44	2.47	3.36	3.24	2.82	3.08	2.71	3.06	9.21
R	2.66	2.41	2.16	2.10	3.85	2.57	3.04	2.71	3.01	3.00	2.53	2.26	2.53	2.94	3.59	2.68	2.46	2.49	2.20	2.83	9.21
S	2.91	2.56	2.40	3.10	2.37	2.65	2.71	2.75	3.18	2.84	2.93	2.69	3.44	2.88	2.74	2.43	2.27	3.08	2.90	2.68	9.21
T	2.66	2.27	2.27	2.17	2.28	2.65	2.26	2.53	2.40	2.67	2.78	2.04	2.53	1.99	2.05	1.79	1.84	2.17	3.81	3.13	9.21
V	2.15	2.56	3.01	2.64	2.69	2.74	2.40	2.38	2.68	2.04	2.64	2.96	2.19	2.43	2.27	2.79	2.36	2.17	2.31	2.37	9.21
W	4.85	3.66	4.11	4.20	4.07	4.04	4.65	4.17	4.07	4.19	4.72	3.87	3.44	3.86	3.77	4.40	5.79	4.10	3.81	4.38	9.21
Y	2.60	2.56	3.29	3.43	2.97	2.65	2.57	3.17	2.63	3.09	4.03	2.69	2.06	2.94	3.13	2.92	3.85	2.90	3.12	2.44	9.21
X	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21